

Abstract. The integration of knowledge extracted from different models described by domain experts or from models generated by machine learning algorithms is strongly conditioned by the lack of an appropriated framework to specify and integrate structures, learning processes, data transformations and data models or data rules. In this work we extended algebraic specification methods to be used in this type of framework. This methodology uses graphic structures similar to Ehresmann's sketches [11] interpreted on a fuzzy set universe. This approach takes advantages of the sketches ability to integrate data deterministic and nondeterministic structures. Selecting this strategy we also try to take advantage on how the graphic languages, used in Category theory in general and used for sketch definition in particular, are suited to reasoning about problems, to structural description and to task specification and task decomposition.

Categorical semiotics

Carlos Leandro

`miguel.melro.leandro@gmail.com`

Departamento de Matemática, Instituto Superior de Engenharia de Lisboa, Portugal.

Introduction

A model is a system of sets with relations providing constraints upon the set system. A class of models, all similarly structured, together with the structure preserving maps between them is a category. For instance a relational database schema can be viewed as a specification of a class of systems of sets having the category of models defined by all the database states and transformations between states. The relational database schema give constraints on the state of the database.

The core of an information system is a set of databases and sets of data transformations, usually taking the form of workflows. In the modern view database presents an internal model of real world fragment, and the transformations offer different ways for construct views of this fragment and integrate the different aspects of it. A crucial step for the proper information system design is to specify the universe and its views in abstract and formalized terms suitable for semantic refinement, such as be used for low-level system specification, and able to be used on the specification improvement through the introduction of new knowledge about the data stored on the system during its life time. This type of data structure specification is called semantic modeling. It has to compress information and process description into a comprehensible way suitable for communication between database tools or designs, such as between data mining processes and data analysts. The usual choice is to use graphic languages, and indeed a great effort has been put in the development of graphic denotational systems. The history of graphic notations invented in various scientific and engineering disciplines is rich. In last years one can observe a great diversity of visual modeling languages and methods: ERdiagrams and a lot of their dialects, OOA&D-schemas in a million of versions and UML which itself comprises a host of various notations. Our goal is to clarify the basic semantic foundations of graph languages and present an integrated framework where different languages and its semantics can be approached consistently and integrated.

A good graphic language should be formalizable, sufficiently expressive to capture all the peculiarities of the real world, and must be suitable for semantic refinement. We are particularly interested in use the same language to model both deterministic and nondeterministic involved structures; for instance data structures and models generated using machine learning algorithms. In our opinion the best approach in terms of expressiveness and formalization to deterministic graphic specification is Category theory.

Category theory generalized the use of graphic language to specify structures and properties through diagrams. These categorical techniques provide powerful tools for formal specification, structuring, model construction, and formal verification for a wide range of systems, presented on a grate variety of papers. The data specification requires finite, effective and comprehensive presentation of complete structures, this type of methodology was explored on Category theory for algebraic specification by Ehresmann. He developed sketches as a specification methodology of mathematical structures and presented it as an alternative to the string-based specification employed in mathematical logic. The functional semantic of sketches is sound in the informal sense that it preserves by definition the structure given in the sketch. The analogy to the semantics of traditional model theory is close enough that sketches and their models fit the definition of "institution" (see [13]), which is an abstract notion of a logic system having syntactic and semantic components. The soundness of semantics appears trivial contrasting with the inductive proof of soundness that occur in string-based logic because the semantics functor is not defined recursively. Sketch specification enjoy a unique combination of rigor, expressiveness and comprehensibility. They can be used for data modeling, process modeling and metadata modeling as well thus providing a unified specification framework for system modeling. However sketch structure forces us to take a global perspective of the system. It makes impossible decomposing a specification problem in subproblems. This makes difficult to specify a large system as the interaction between subsystems or components, which is a common practice, in engineering. We can give a better view to this problem by means of a typical application, the specification of workflows (for more details see [20]). A workflow describes a business process in terms of tasks and shared resources. Such descriptions are needed, for example, when interoperability of the workflows of different organizations is an issue, for example, when applications of different enterprises are to be integrated over the Internet [20]. A workflow is a net [30], some times a Petri net, satisfying some structural constraints and the corresponding soundness conditions. Usually the methodology used to specify workflows have associated a library of components. An interorganizational workflow is modeled as a set of such workflow nets [31] connected through additional places for asynchronous communication and synchronisation requirements on transitions. The difficulty of applying sketch to this type of task results of the way composition is defined on the the category of sets. We want to interpret a workflow as a arrow, where the gluing of different workflows must be also interpreted as an arrow and must be always defined. The net structure defined by a workflow is called an oriented multi-graph since its components are relations linking together families defined by sets of entities or data structures.

For our goal we extend the syntax of sketch to multi-graphs and define its models on a class of fuzzy relations (see [14]). Where multi-graphs nodes are interpreted as relations. To extend the syntax of a sketch we began by formalize a library as the syntactic structure of admissible configurations using components on that library. This approach is based on the notion of component, used to

define relationships between two families of entities, and which can be organized following an hierarchy of complexity. A component if not atomic is defined by the plugging of other components. We see the set of admissible configurations as the graphic language defined by the library structure. A model of a library is a map from the library multi-graph to the structure associated to the class of relations defined using a multi-valued logic. An interpretation for an admissible configuration, is defined for each library model. It is the limit of the multi-graph, in the category of Ω -sets, resulting of applying a library model to a admissible multi-graph. It seems to be the adequate framework for the definition and the study of graphic-based logics, if we interpret one of its node as the set of truth values.

We used libraries as a way to define the lexicon of the language used on the description of a domain. The category defined by library models and natural transformation aren't a accessible category (see [11]) it can't be axiomatized by a basic theory in first-order logic. By this we mean what classic Ehresmann sketches not have sufficient expressive power to specify the category of library models defined in a multi-valued logic with more than three truth values.

To be able to formalize linguistic, Chomsky in [5] propose a language as a set of grammatically correct sentences possible in the language. The goal of defining a language is then to characterize the set of grammatical sentences explicitly, by means of a formal grammar. The two main categories of grammar are that of *generative grammar*, which are sets of rules for how elements of a language can be generated, and that of *analytic grammars*, which are sets of rules for how a structure can be analyzed to determine whether it is a member of the language. In this sense, our approach to the definition of a graphic language based on libraries of components uses a multi-graph as the language analytic grammar.

A generative grammar does not in any way correspond to the algorithm used to parse the generated language. Analytic grammar corresponds more directly to the structure and the semantic of a parser for the language. Examples of analytic grammars formalisms include top-down parsing language (TDPL)[1], link grammars [7] and parsing expression grammars [9].

Our extension to the syntax of sketch is based on Link grammars. A theory of syntax proposed by Davy Temperley and Daniel Slator in [7] which builds relations between pairs of words, rather than constructing constituents in a tree-like hierarchy. They are similar to dependency grammars developed by Lucien Tesnière in the 60's [28]. Link grammar is a form of analytic grammar designed for linguistics, which derives syntactic structures by examining the positional relationships between pairs of words. This can be seen in the model that the Davy Temperley and Daniel Slator provided for English in this system: Their grammar deals with most of the linguistic phenomena in English. Informally, a sentence is correct in this system if it is possible to link all words according to the links needed by each word, defined in the lexicon. Link represents syntactic relations. It is shown in [7] that link grammars have the expressive power of context-free grammars.

We call sign system to the extension of the Ehresmann sketch notion. A formal definition for an identical notion appeared is presented by Goguen in [10], in our version it is a library defined by a multi-graph more, just as sketch, a set of commutative multi-diagrams, a set of limit cones and a set of colimit cocones. In this context the limit and the colimit for a multi-diagram must be seen as a relation defined on a fuzzy set theory. We consider a semiotic system as a pair defined by a sign system and one of its models, and they can be seen as an institution in Goguen sense [13].

Our goal was to develop a mathematically precise theory of semiotics based on Ehresmann sketch notion. This structure try to internalized the formalization made by the *Vienna Circle* in their *International Encyclopedia of Unified Science*, by breaking out the field, which they called "*Semiotic*", into three branches: *Semantics* (relation between signs and the things they refer to) *Syntactics* (Relation of signs to each other in formal structures) and *Pragmatics* (Relation of signs to their impacts on those processes which use them).

Signs appear as members of sign systems. Must signs are complex objects constructed from others, lower level signs. In this sense they can be seen as structures defined using signs and sign systems capture the systematic structure of signs. Marking an entity with a sign can be seen as a way to named entities or assign properties to entities, such as two entities marked with the some signs are identical. We defined the model of a sign system a consistent process for marking entities belonging to a fixed universe, preserving labeled relations between signs.

Following the spirit used by Peirce on *Semiotics* we identify entities from the universe of discourse with some of this signs. We do this by marking some of the objects or morphisms of a topos with signs used on library specification using a library model. If, in the topos, the object classifier have a monoidal logics structure, and its operators are marked with signs from a library, we can use this library to specify monoidal graphic logics. And a relation on a semiotic system must be seen as a configuration having by interpretation a multi-morphism with target a object marked as having associated a monoidal logic structure. This allows the extension of the concepts used in logics to the graphic logics associated to a semiotic system. We can use graphic relations to define queries on the semiotic. An answer for it is a fibre on the source, of its interpretation, defined by all the "points" transformed in the true for the semiotic associated ML-algebra. In this context a "set" of points is consistent with a graphic relation if every point is associated by the relation interpretation to the true of the semiotic monoidal logic. Since monoidal logics can be seen as fuzzy logics or multi-valued logics, logics definable in this framework are in reality fuzzy graphic logics. This extends the logic used for Ehresmann sketches. And allows making fuzzy the notion of a relation evaluation, an important issue for practical applications.

Day living activities generate information which can be stored in information systems spread by different databases. A query to a information system can be seen as a view of the data stored in the system, and it is presented by a dataset. Many times this information is useful to produce new knowledge about the real-

ity. Given the amount of data stored this transformation must be automatized and this is the goal of fields of AI, like Machine Learning, see [27].

In fuzzy set theory a dataset can be expressed by a relation. The interpretation of a word in a logic semiotic is called a model for the dataset if the relation is the multi-diagram limit. If we see a dataset as a way of codifying data, we can see its model as a way of represent the knowledge in the graphic language, associated to a fixed logic semiotic. This relation between data and knowledge, is only useful, if we define the notion of structure λ -consistent with a relation. Where λ is a logic value quantifying the degree of similarity between interpretation of a word and the concept defined codified in the dataset. In this sense the fact of a dataset be λ -consistent with a diagram catches the idea of approximation.

If the data, presented in a dataset, is λ -consistent with a set of diagrams we call semiotic defined by this set of diagrams a theory λ -consistent with the data. Naturally this can be extended to databases. The knowledge available about a database can be codified in a semiotic and the quality of this knowledge is given by the way its model describe good approximations to data, associated to tables what can be, generated in a database state. Since different human specialists or machine learning algorithms express the extracted knowledge using, many times, different languages: the problem of knowledge integration is the problem of semiotic integration. The objective of integration is then to construct one semiotic that exploits all the knowledge that is available and as good performance. We describe methodologies for merging several separate theories. However this processes some times also requires the integration of languages and the associated logics which is simplified following our approach.

Overview of the paper:

We began section 1 describing a partially-ordered monoidal structure for the set of truth-values used on the definition of our graphic logics. The language used in this logics is based on possible circuit configurations using a libraries of components, structure defined on section 5. As a framework for the definition of models for this libraries we used a class of relations defined between sets and evaluated in a multi-valued logic, which are described in section 2. For that we need to define composition of relations compatible with circuit gluing. In this sense composition must be seen as a total operator in the class of relations, relaxing the diagram equality, evaluated in a multi-valued logic, allowing in section 4, the presentation of a generalized version for the notion of commutative diagrams. This is explored in section 3, on the definition of a version for Bayesian inference on fuzzy logics. In section 6 the language defined by a libraries is seen as a set of circuits closed to the plug-in operation, every word is a string of component labels or signs and define a relation between a family of inputs requirements and a family of output structures, both identified using families of signs. We present how libraries are modeled on the class of relations on section 6. The descriptive power of languages defined using libraries allow defining structure what are not definable using first order basic theory. This is presented in section 7 by showing what the category defined using library models is not accessible. Accessible cat-

egories are known to be specified using Ehresmann sketch. We take advantage of its specification power by enriching the structure of a library with a structure similar to a Ehresmann sketch, in section 8, we called to this specification tool a sign system or a specification system. This enrichment is made using multi-diagrams instead of diagrams specified respecting library constraints and where limits and colimits are interpreted as multi-morphisms and used in section 4, on the definition of diagram commutativity evaluation on a multi-valued logic. A specification system where we fixed a model we called a semiotic system, with this and an example we finish section 8. On section 9 we use sign systems to specify fuzzy logics. They are semiotics with special structure, for that we impose interpretations for some of the sign system signs allowing the interpretation of words as evaluations of relations in a monoidal logic. When some signs are interpreted as ML-algebras operators the library was called a logic library and the associated language was called a logic language. We formalize this concept and describe when a diagram defines a relation and an equation. However, some problems in Mathematics require more expressive languages than the ones defined using libraries. We improve the expressive power of libraries Lagrangian syntactic operators. An example is described on section 10 allowing the definition of Differential semiotics. On section 11 we present how to evaluate relations in a semiotic, and use it to define what we mean by the level of consistence of a relation. This notion is extended to relations λ -consistent with words in a semiotic. We emphasize the idea that a diagram defining a relation can be seen as a query to the semiotic. In this context a λ -answer is a structure where the query is λ -consistent. These notions are used on section 12 the definition of bottom and upper presentation to a structure A in the semiotic; the bottom presentation is the lower structure in A codified in the semiotic language, and the upper presentation is the short structure containing A and codified in the fixed language. These notions can be seen as two approximations to the concept, following the spirit of Pawlak's of the standard version of rough set theory. They define an interior and a closure operators for structures, allowing the definition of a formal topology. Which we use on the definition of an inference system for words evaluation on a semiotic system based on properties of ML-algebras. What is made for descriptions can be made for relations evaluated in a multi-valued logic. In section 12 a fuzzy relation is computable in a semiotic if it can be seen as an interpretation for a diagram defined in the associated language. Section 14 is dedicated to the integration of semiotics and models for concepts. The goal is to construct one system that exploits all the knowledge that is available, allowing to improve concept description by combining different descriptions for the concept on the same concept possibility expressed using different languages. For specification reasons a string-based modal logic is present, on section 15, where propositional variables are interpreted as diagrams defined on a semiotic. This intends to be a meta-language to reasoning about models of concepts and knowledge.

1 Monoidal logics

Fuzziness is the rule than the exception in practical problems. A lot of research is being done on fuzzy sets and the associated fuzzy logics; we are specially interested in the possibility of extending the data specification paradigm using Ehresmann sketches to the fuzzy case. This paper is motivated to the introduction to Ω -Categories given in [19] and to Ω -Sets given in [29].

Ulrich Höhle introduced Monoidal Logic in 1995 in order to give a common framework to several first order non-classical logics, such as Linear logic, Intuitionistic logic and Lukasiewicz logic. A Monoidal Logic is a Full Lambek calculus with exchange and weakening. We supposed what problems involving the specification of structures, using libraries of components, can be formulate in a framework defined by a set theory with a monoidal logic.

Recall that a algebra $(\Omega, \otimes, \leq, 1)$ is a partially-ordered monoid if $(\Omega, \otimes, 1)$ is a monoid and \leq is a partial order on Ω such that the operator \otimes is monotone increasing; i.e.

$$x \leq x' \text{ and } y \leq y' \text{ imply } x \otimes y \leq x' \otimes y'.$$

An algebra $(\Omega, \otimes, \backslash, /, \leq, 1)$ is a resituated partially-ordered monoid if $(\Omega, \otimes, \leq, 1)$ is a partially-ordered monoid and moreover the following condition is satisfied for all $x, y, z \in \Omega$;

$$x \otimes y \leq z \Leftrightarrow y \leq x \backslash z \Leftrightarrow x \leq z / y.$$

This condition is called the *law of residuation*, and $/$ and \backslash are called the right and left residual of \otimes , respectively.

Any residuated partial-ordered monoid Ω such that (Ω, \leq) forms a lattice and (Ω, \otimes) has a unit it is called a *residuated lattice*. More precisely, an algebra

$$(\Omega, \vee, \wedge, \otimes, \backslash, /, 1)$$

is a residuated lattice if

1. $(\Omega, \otimes, 1)$ is a monoid such that \backslash and $/$ are the right and the left residual of \otimes , respectively, and
2. (Ω, \vee, \wedge) is a lattice.

When \otimes is commutative, we call it a *commutative residuated lattice*. In any commutative residuated lattice, $x \backslash y = y/x$ hold for all x, y . In such a case, we use the symbol \Rightarrow and write $x \Rightarrow y$ instead of $x \backslash y$ (and of y/x). Also the commutative residuated lattice is denoted by $(\Omega, \vee, \wedge, \otimes, \Rightarrow, 1)$.

Definition 1 A ML-algebra is a bounded commutative residuated lattice where $1 = \top$, formally, is a system $(\Omega, \otimes, \Rightarrow, \vee, \wedge, \perp, \top)$ satisfying:

1. (Ω, \otimes, \top) is a commutative monoid,
2. $x \otimes \top = x$ for every $x \in \Omega$,
3. $(\Omega, \vee, \wedge, \perp, \top)$ is a bounded lattice, and
4. the residuation property holds,

$$\text{for all } x, y, z \in \Omega, x \leq y \Rightarrow z \text{ iff } x \otimes y \leq z.$$

In this paper we assume that ML-algebra Ω is non-trivial, i.e. $\top \neq \perp$.

A structure equivalent to a ML-algebra is presented in [19] as a commutative and unital quantale where Ω is a complete lattice equipped with a symmetric and associative tensor product \otimes , with unit \top and with right adjoint \Rightarrow . Considered Ω as a thin category, Ω is said to be symmetric monoidal-closed.

Logics having as models refinements of ML-algebras are called *monoidal logics*. In many-valued logics, such as fuzzy logics, \otimes is the standard truth degree function for conjunction connective. Since operator \otimes is monotone and have right adjoint, we have:

Proposition 1 *On a ML-algebra one has*

1. if $y \leq z$ then $x \otimes y \leq x \otimes z$,
2. $x \leq y$ iff $(x \Rightarrow y) = \top$, and
3. $x \Rightarrow z = \bigvee \{y : x \otimes y \leq z\}$.

And,

Proposition 2 [8] *In any ML-algebra the following equalities hold, for all $x, y, z \in \Omega$,*

1. $x \otimes (x \Rightarrow y) \leq x \wedge y$, and
2. $(x \Rightarrow y) \otimes (y \Rightarrow z) \leq x \Rightarrow z$.

Every non-trivial Heyting algebra - with $\otimes = \wedge$ and \top the top element - is an example of a ML-algebra, in particular the two element chain $2 = \{false < true\}$ with the monoidal structure given by "and" and "true".

The complete real half-line $P = [0, \infty]$, with the categorical structure induced by the relation \leq admits several interesting monoidal structures. If $\otimes = \wedge = \max$ it is a Heyting algebra. Another possible choice of \otimes is $+$, note that in this case the right adjoint \Rightarrow is given by truncated minus: $x \Rightarrow y = \max\{v - u, 0\}$.

Example 1 (t-norm based fuzzy logic) *A t-norm is a function \otimes used to define a ML-algebra structure on the real unit interval $[0, 1]$. We may define a monoidal logic using a t-norm by taken the unite interval as the set of truth values and where the residuum of \otimes is defined as the operation $x \Rightarrow y = \max\{z | x \otimes z \leq y\}$. The other truth function considered important in fuzzy logic are weak conjunction $x \cdot y = \min(x, y)$ and weak disjunction $x + y = \max(x, y)$. However in the following we interpret a fuzzy logic on a ML-algebra $([0, 1], \otimes, \Rightarrow, \vee, \wedge, 0, 1)$ where \otimes is continuo t-norm and \Rightarrow is its residuum, the lattice structures are given by $x \vee y = \max(x, y)$ and $x \wedge y = \min(x, y)$. The following are importante examples of fuzzy logics interpreted in ML-algebras defined by specific continuous t-norms:*

- **Lukasiewicz** logic defined using the t-norm $x \otimes y = \max(0, x + y - 1)$ and its residuum

$$x \Rightarrow y = \min(1, 1 - x + y)$$

- **Gödel logic** defined using the t -norm $x \otimes y = \min(x, y)$ and its residuum

$$x \Rightarrow y = \begin{cases} 1 & \text{if } x \leq y \\ y & \text{otherwise} \end{cases}$$

- **Product logic** defined using the t -norm $x \otimes y = x.y$ and its residuum

$$x \Rightarrow y = \begin{cases} 1 & \text{if } x \leq y \\ y/x & \text{otherwise} \end{cases}$$

Particularly important to this work are the basic logics, with have by instances ML-algebras with are divisible, i.e. such that

$$x \otimes (x \Rightarrow y) = x \wedge y.$$

Examples of this type of logic are classic boolean logic and fuzzy logic like product, Gödel and Łukasiewicz. Note that must of the examples presented in the following are construct using product logics with the natural order in interval $[0,1]$.

For the sequel we define

$$a \Leftrightarrow b := (a \Rightarrow b) \otimes (b \Rightarrow a) \text{ and } \neg a := a \Rightarrow \perp.$$

Let

$$(\Omega_i, \otimes_i, \Rightarrow_i, \vee_i, \wedge_i, \perp_i, \top_i)_{i \in I}$$

be a finite family of ML-algebra. The product of this ML-algebras is the ML-algebra

$$(\prod_{i \in I} \Omega_i, \otimes, \Rightarrow, \vee, \wedge, \perp, \top)$$

such that

1. $\prod_{i \in I} \Omega_i$ is the cartesian product of sets of truth values,
2. $(\lambda_1, \lambda_2, \dots, \lambda_n) \otimes (\alpha_1, \alpha_2, \dots, \alpha_n) = (\lambda_1 \otimes \alpha_1, \lambda_2 \otimes \alpha_2, \dots, \lambda_n \otimes \alpha_n),$
3. $(\lambda_1, \lambda_2, \dots, \lambda_n) \Rightarrow (\alpha_1, \alpha_2, \dots, \alpha_n) = (\lambda_1 \Rightarrow \alpha_1, \lambda_2 \Rightarrow \alpha_2, \dots, \lambda_n \Rightarrow \alpha_n),$
4. $(\lambda_1, \lambda_2, \dots, \lambda_n) \vee (\alpha_1, \alpha_2, \dots, \alpha_n) = (\lambda_1 \vee \alpha_1, \lambda_2 \vee \alpha_2, \dots, \lambda_n \vee \alpha_n),$
5. $(\lambda_1, \lambda_2, \dots, \lambda_n) \wedge (\alpha_1, \alpha_2, \dots, \alpha_n) = (\lambda_1 \wedge \alpha_1, \lambda_2 \wedge \alpha_2, \dots, \lambda_n \wedge \alpha_n),$
6. $\perp = (\perp_1, \perp_2, \dots, \perp_n),$ and
7. $\top = (\top_1, \top_2, \dots, \top_n).$

This structure has associated two types of morphisms. The projections

$$\pi_j : \prod_{i \in I} \Omega_i \rightarrow \Omega_j,$$

and the upper interpretations

$$\begin{aligned} \top_j : \Omega_j &\rightarrow \prod_{i \in I} \Omega_i \\ \alpha_j &\mapsto (\perp_1, \perp_2, \dots, \alpha_j, \dots, \perp_n) \end{aligned}$$

Note what upper interpretation is the right inverse to projection,

$$\pi_j \cdot \top_j = id_{\Omega_j}.$$

We will use this structure as a vehicle for integration of ML-logics.

2 Multi-morphisms

A surprising result discovered in Category Theory, presented by M. Makkai in [16], is that the arrow specification language is absolutely expressive, in the sense that any construction having a formal semantic meaning can be described in the arrow language as well. Moreover, if basic object of interest are described by arrows then normally it turned out that many derived objects of interest can be also derived by arrows in a quite natural way [32]. To define the universe we are going to deal with it is necessary and sufficient to define what we mean by a morphisms between objects of the universe.

Our universe for semantic modeling must be "essentially the same" as *Set* but able to represent soft structures specified thought monoidal logics. Let Ω be a set with a ML-algebra structure $(\Omega, \otimes, \Rightarrow, \vee, \wedge, \perp, \top)$. We use as universe $Set(\Omega)$ defined using *Set* having by entities Ω -sets, i.e. sets A furnished with a Ω -valued map

$$[\cdot = \cdot] : A \times A \rightarrow \Omega,$$

which is symmetric and transitive in the sense that both

$$[a = b] = [b = a] \text{ and } [a = b] \otimes [b = c] \leq [a = c],$$

hold for all $a, b, c \in A$. This is called a *similarity* in A . We will use Greek letters to denote Ω -sets, we write $\alpha : A$, to mean a Ω -sets defined by set A and a similarity $[\cdot = \cdot]_\alpha$, and it is interpreted as a relation evaluated in Ω or a distribution in $A \times A$. The diagonal of this fuzzy relation is used on definition of fuzzy sets with support A . For each Ω -set $\alpha : A$ and $a \in A$ we define

$$[a]_\alpha = [a = a]_\alpha,$$

and called it the *extend* of a . Then $[\cdot]_\alpha : A \rightarrow \Omega$ is a representation for the fuzzy set α codified thought similarity $[\cdot = \cdot]_\alpha$. An element a is called *global* in $\alpha : A$ if $[a]_\alpha = \top$.

Note that every set A have a natural structure of Ω -set defined by the equality $=$ in A , i.e. having by similarity

$$[a = b]_A = \begin{cases} \top & \text{if } a = b \\ \perp & \text{if } a \neq b \end{cases}.$$

The crisp similarity $[a = b]_A$, defined by the equality in A , is denoted by 1_A .

Entities belonging to a Ω -set $\alpha : A$ are characterized by a set of attributes $(A_i)_I$ if $A = \prod_{i \in I} A_i$. Given $\bar{x} \in \prod_{i \in I} A_i$, on the description of \bar{x} many of the values associated to some of this attributes are "non-observable" or unknown. In this sense we will differentiate between two types of attributes: *observable attributes* and *non-observable attributes*. Let $(A_i)_{i \in L}$ be a set of observable attributes in A , where $L \subseteq I$. We define an observable Ω -set of $\alpha : A = \prod_{i \in I} A_i$ as the Ω -set $\beta : B = \prod_{i \in L} A_i$ such that

$$[\bar{a} = \bar{b}]_\beta = \bigvee_{\bar{x}=(\bar{c},\bar{a}),\bar{y}=(\bar{d},\bar{b}) \in A} [\bar{x} = \bar{y}]_\alpha.$$

Definition 2 (Observable description) If $\alpha : A$ is a Ω -set with a set of observable attributes $(A_i)_{i \in L}$, we call to every $\bar{a} \in \prod_{i \in L} A_i$ an observable description for an entity in α .

We define a multi-morphism in $Set(\Omega)$ as a tracking morphism between Ω -sets $\alpha : A$ and $\beta : B$ as a map

$$f : A \times B \rightarrow \Omega$$

(usually called a Ω -map or a Ω -matrix in [19]). If f is a multi-morphism between $\alpha : A$ and $\beta : B$ in $Set(\Omega)$ we write $f : A \multimap B$ to identify A as the source of f and B as the target of f . And if \bar{a} and \bar{b} are observable descriptions for entities in $\alpha : A$ and $\beta : B$ respectively we define

$$f(\bar{a}, \bar{b}) = \bigvee_{\bar{x}=(\bar{c}, \bar{a}), \bar{y}=(\bar{d}, \bar{b})} f(\bar{x}, \bar{y}).$$

The complete partial order on the ML-algebra Ω induces a complete partial order on the set of multi-morphisms. Given two multi-morphisms between Ω -sets $\alpha : A$ and $\beta : B$ in $Set(\Omega)$

$$f, g : A \times B \rightarrow \Omega$$

we write $f \leq g$ if $f(a, b) \leq g(a, b)$, for every $(a, b) \in A \times B$. Graphically a multi-morphism

$$f : A_0 \times A_1 \times A_2 \multimap A_3 \times A_4 \times A_5$$

is present in fig. 1 by a multi-arrow having by sources A_0, A_1 and A_2 and by

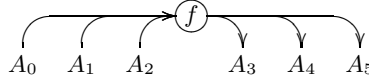


Fig. 1. Multi-arrow.

targets A_3, A_4 and A_5 .

We classify multi-morphisms preserve entity evaluation in Ω :

Definition 3 (Total multi-morphism) A multi-morphism $f : A \multimap B$ is total in $\alpha : A$ if

$$[a]_\alpha = \bigvee_b f(a, b),$$

for every $a \in A$.

Definition 4 (Faithful multi-morphism) A multi-morphism $f : A \multimap B$ is faithful in $\beta : B$ if

$$[b]_\beta = \bigvee_a f(a, b),$$

for every $b \in B$.

Note what, for every Ω -set $\alpha : \prod_{i \in I} A_i$ we can use the similarity diagonal to define a multi-morphism by selecting a set of sources $(A_s)_{s \in S}$ sets and a set of targets $(A_t)_{t \in T}$ sets, with disjoint indexes, i.e. such that $S \cap T = \emptyset$. This multi-morphism is given by a map

$$g(\bar{x}, \bar{y}, \bar{z}) = \bigvee_{\bar{y} \in I \setminus (S \cup T)} [\bar{x}, \bar{y}, \bar{z}]_{\prod_{i \in I} A_i},$$

for every $\bar{x} \in \prod_{s \in S} A_s$ and $\bar{z} \in \prod_{t \in T} A_t$, which defines

$$g : \prod_{s \in S} A_s \multimap \prod_{t \in T} A_t.$$

Composition of multi-morphisms is defined as matrix multiplication, the composition of

$$f : A \multimap B \text{ and } g : B \multimap C,$$

as the multi-morphism

$$f \otimes g : A \multimap B,$$

given by

$$(f \otimes g)(a, c) = \bigvee_b (f(a, b) \otimes g(b, c)).$$

Note what if f and g are total and faithful then $f \otimes g$ is total and faithful. This composition have by identity for a Ω -set $\alpha : A$ the multi-morphism

$$1_A = [\cdot = \cdot]_\alpha : A \multimap A,$$

defined by the equality in A , since for $f : A \multimap B$ we have $1_A \otimes f = f \otimes 1_B$.

Proposition 3 *In $Set(\Omega)$ let $f : A \multimap A$ be a multi-morphism such that $1_A \leq f$. If in the ML-algebra Ω for every truth value α , $\alpha \otimes \alpha \leq \alpha$, then*

$$1_A \leq f \otimes f \leq f.$$

And, when the logic have more than two truth values, i.e. if $|\Omega| > 2$, we have

$$f \otimes f = 1_A \text{ iff } f = 1_A.$$

The set of multi-morphisms defined between Ω -sets $\alpha : A$ and $\beta : B$ is denoted by $Set(\Omega)|A, B|$. And, every map $f : A \rightarrow B$ in Set defines a multi-morphism, with source A and target B , given by

$$\chi_f : A \times B \rightarrow \Omega \text{ where } \chi_f(a, b) = \begin{cases} \top & \text{if } f(a) = b \\ \perp & \text{if } f(a) \neq b \end{cases}.$$

In this sense the hom-set $Set[A, B]$, of morphism between A and B in Set , define a subset of $Set(\Omega)|A, B|$. To keep notation simple in the sequel we will write $f : A \rightarrow B$ rather than $f : A \multimap B$ for the multi-morphism induced by a map.

Then $f : A \rightarrow B$ defines a total multi-morphism from $\alpha : A$, having by similarity the equality, and it is a faithful multi-morphism to $\beta : B$, having by similarity

$$[a = b] = \begin{cases} \top & \text{if } a = b \text{ and } a \in f(A) \\ \perp & \text{other wise} \end{cases}.$$

The formula for multi-morphism composition became considerably easier if one of the multi-morphism is a set-map. For maps $f : A \rightarrow B$ and $g : B \rightarrow C$ and multi-morphisms $r : A \rightarrow B$ and $s : B \rightarrow C$ we have

$$(f \otimes s)(a, c) = s(f(a), c), \text{ and } (r \otimes g)(a, c) = \bigvee_{b \in g^{-1}(c)} r(a, b).$$

The operator of multi-morphism composition can be extended to multi-morphisms which are not composable, in the usual sense. Let

$$f : A \rightarrow X \times W \text{ and } g : B \times X \rightarrow C,$$

then we define

$$f \otimes g : A \times B \rightarrow W \times C,$$

given by

$$(f \otimes g)(a, b, w, c) = \bigvee_x (f(a, x, w) \otimes g(b, x, c)).$$

In particular if $f : A \rightarrow B$, $g : C \rightarrow D$ and $B \neq C$ then

$$f \otimes g : A \times C \rightarrow B \times D,$$

is given by

$$(f \otimes g)(a, c, b, d) = f(a, b) \otimes g(c, d).$$

This reflects the independence between entities in B and C and we define:

Definition 5 (Independence) *Two multi-morphisms f and g are called independent if*

$$f \otimes g = g \otimes f.$$

Example 2 (Keys in a relational database) *The relational model for database management is a database model based on predicate logic and set theory. The fundamental assumption of the relational model is that data is represented as mathematical n -ary relations, an n -ary relation being a subset of the Cartesian product of n domains. In the usual mathematical model, reasoning about such data is done in two-valued logic or three-valued logic. Data are operated upon by means of a relational calculus or relational algebra.*

The relational model of data permits the database designer to create a consistent, logical representation of information. Consistency is achieved by including declared constraints in the database design, which is usually referred to as the logical schema.

A weight table R in a database defined using attributes $(A_i)_I$ is a map in a ML-algebra

$$R : \prod_{i \in I} A_i \rightarrow \Omega.$$

in this sense a weight table is a Ω -set $\alpha : \prod_{i \in I} A_i$.

Every weight table $R : A \times B \rightarrow \Omega$, may be describe as the multi-morphism $R : A \multimap B$, and can be decomposed using two weight tables

$$D_0 : A \times K \rightarrow \Omega$$

$$D_1 : K \times B \rightarrow \Omega$$

such that

$$R = D_0 \otimes D_1.$$

In this case we call to K a set of keys, between D_0 and D_1 , and write

$$D_0 \otimes_K D_1$$

to denote the joint of D_0 and D_1 using the keys in K .

Generically, if K_1, K_2, \dots, K_n are sets of keys between D_0 and respectively D_1, D_2, \dots, D_n we write

$$D = D_0 \otimes_{K_1, K_2, \dots, K_n} (D_1 \otimes \dots \otimes D_n),$$

to denote the joint product

$$D = (\dots (((D_0 \otimes_{K_1} D_1) \otimes_{K_2} D_2) \otimes_{K_3} \dots) \otimes_{K_n} D_n),$$

or

$$D = D_0 \otimes_{K_1} D_1 \otimes_{K_2} D_2 \otimes_{K_3} \dots \otimes_{K_n} D_n.$$

When the family (K_i) of keys is defined by the same set K the joint product is called the K indexed product of D_0, D_1, \dots, D_n and denoted by

$$D = D_0 \otimes_K D_1 \otimes_K D_2 \otimes_K \dots \otimes_K D_n.$$

In this case D is called the K -indexed product of $D_0, D_1, D_2, \dots, D_n$.

Given the importance of multi-morphism composition in this work lets formalize that we mean by the multi-morphism composition:

Definition 6 (Multi-morphism composition) Given multi-morphisms f and g defined by Ω -maps

$$f : \prod_{i \in I(f)} A_i \rightarrow \Omega \text{ and } g : \prod_{j \in I(g)} B_j \rightarrow \Omega$$

where for every $i \in I(f)$ and $j \in I(g)$, $i = j$ iff $A_i = B_j$. Without selection of sources and targets sets for f and g we define

$$(f \otimes g)(\bar{x}, \bar{y}) = f(\bar{x}) \otimes g(\bar{y}),$$

for every $\bar{x} \in \prod_{i \in I(f)} A_i$ and $\bar{y} \in \prod_{j \in I(g)} B_j$. However, if we select sets of sources $S(f) \subset I(f)$ and $S(g) \subset I(g)$, and sets of targets $T(f) \subset I(f)$ and $T(g) \subset I(g)$, such $S(f) \cap T(f) = \emptyset$ and $S(g) \cap T(g) = \emptyset$, we define

$$(f \otimes g)(\bar{x}, \bar{y}) = \bigvee_{\bar{z} \in \prod_{i \in T(f) \cap S(g)} A_i} f(\bar{x}, \bar{z}) \otimes g(\bar{z}, \bar{y}),$$

for every $\bar{x} \in \prod_{i \in S(f)} A_i \times \prod_{j \in S(g) \setminus T(f)} B_j$ and $\bar{y} \in \prod_{i \in T(f) \setminus S(g)} A_i \times \prod_{j \in T(g)} B_j$.

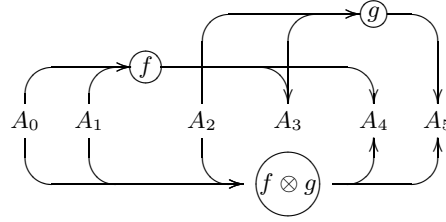


Fig. 2. Multi-morphism composition.

The transpose $f^\circ : B \multimap A$ of a multi-morphism $f : A \multimap B$ is defined by $f^\circ(b, a) = f(a, b)$. It is easy to see that

$$(\cdot)^\circ : \text{Set}(\Omega)|A, B| \rightarrow \text{Set}(\Omega)|B, A|$$

is order preserving and

$$[\cdot = \cdot]^\circ = [\cdot = \cdot], \quad (f \otimes g)^\circ = g^\circ \otimes f^\circ \text{ and } f^{\circ\circ} = f.$$

In this sense, if f is a multi-morphism having by set of sources \mathcal{A} and by set of targets \mathcal{B} then \mathcal{B} is the set of sources for f° and \mathcal{A} its set of targets.

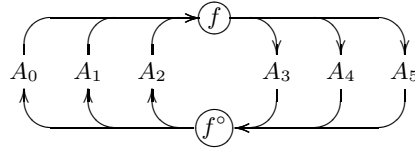


Fig. 3. Transpose.

We classify multi-morphisms by its ability to preserve its domain or codomain truth values distribution.

Definition 7 A multi-morphism $f : A \multimap B$ is an epimorphism between $\alpha : A$ and $\beta : B$ if

$$f^\circ \otimes \alpha \otimes f = \beta.$$

It is a monomorphism between $\alpha : A$ and $\beta : B$ when

$$\alpha = f \otimes \beta \otimes f^\circ.$$

Naturally, when the two conditions are valid f is called an isomorphism between Ω -objects $\alpha : A$ and $\beta : B$.

For each set-map $f : A \rightarrow B$ we have

$$[\cdot = \cdot]_A = 1_A \leq f \otimes f^\circ \text{ and } f^\circ \otimes f \leq 1_B = [\cdot = \cdot]_B,$$

i.e. f is left adjoint to f° , $f \dashv f^\circ$. If $f : A \multimap B$ is a multi-morphism and $1_A = f \otimes f^\circ$ and $f^\circ \otimes f = 1_B$ the multi-morphism f is called *orthogonal*.

In general given multi-morphisms $f : A \multimap B$ and $g : B \multimap A$ we say that f is the *left adjoint* to g for $\alpha : A$ and $\beta : B$ if

$$[\cdot = \cdot]_\alpha \leq f \otimes g \text{ and } g \otimes f \leq [\cdot = \cdot]_\beta.$$

The tensor product on Ω can be naturally transported to Ω -sets. More precisely, for Ω -sets $\alpha : A$ and $\beta : B$, we denote by $\alpha \otimes \beta$ the Ω -sets defined using the Cartesian product $A \times B$ in $Set(\Omega)$ and furnished with

$$[(a_1, b_1) = (a_2, b_2)]_{\alpha \otimes \beta} = [a_1 = a_2]_\alpha \otimes [b_1 = b_2]_\beta.$$

Then, for each Ω -set $\alpha : A$, the functor

$$\alpha \otimes \cdot : Set(\Omega) \rightarrow Set(\Omega),$$

has a adjoint the hom functor $(\cdot)^\alpha : Set(\Omega) \rightarrow Set(\Omega)$ defined by $\beta^\alpha = Set(\Omega)[\alpha, \beta]$ with the similarity given by

$$[f = g]_{\beta^\alpha} = \bigwedge_{a \in A} \bigwedge_{b \in B} (f(a, b) \Leftrightarrow g(a, b)).$$

and, for every $f \in Set(\Omega)[\beta, \gamma]$,

$$f^\alpha = Set(\Omega)[\alpha, f] : Set(\Omega)[\alpha, \beta] \rightarrow Set(\Omega)[\alpha, \gamma],$$

such that $(f^\alpha)(g) = g \otimes f$.

Being monoidal-closed Ω has a natural structure as Ω -set given by

$$[x = y]_\Omega = (x \Leftrightarrow y) = (x \Rightarrow y) \otimes (y \Rightarrow x).$$

Given similarities $[\cdot = \cdot]_\alpha : A \multimap A$ and $[\cdot = \cdot]_\beta : B \multimap B$. If we sets A and B are distinct, applying composition definition we have

$$[\cdot = \cdot]_\alpha \otimes [\cdot = \cdot]_\beta : A \times B \multimap A \times B,$$

given by

$$([\cdot = \cdot]_\alpha \otimes [\cdot = \cdot]_\beta)(a_1, b_1, a_2, b_2) = [a_1 = a_2]_\alpha \otimes [b_1 = b_2]_\beta,$$

and it is a similarity relation defining the Ω -object $\alpha \otimes \beta : A \times B$. More generically we define:

Definition 8 (Product of Ω -sets) *Given Ω -sets $\alpha : A$ and $\alpha : B$ we define the product of $\alpha \otimes \beta$ as the Ω -sets $\alpha \otimes \beta : A \times B$ given by*

$$[\cdot = \cdot]_{\alpha \otimes \beta} : A \times B \times A \times B \rightarrow \Omega,$$

such that

$$[(a_1, b_1) = (a_2, b_2)]_{\alpha \otimes \beta} = [a_1 = a_2]_\alpha \otimes [b_1 = b_2]_\beta.$$

By the transitivity imposed on the definition of similarity we have,

$$[\cdot = \cdot]_{\alpha \otimes \beta} = [\cdot = \cdot]_\alpha \otimes [\cdot = \cdot]_\beta \leq [\cdot = \cdot]_\alpha.$$

3 Bayesian inference in a basic logic

The presented definition for multi-morphism composition \otimes is compatible to the Bayes' theorem used on Bayesian inference when $Set(\Omega)$ logic is a basic logic.

Proposition 4 (Bayes Rule) *Let Ω be a divisible ML-algebra. Given a faithful and total multi-morphism $f : A \rightarrow B$, and observable descriptions a and b of entities in $\alpha : A$ and $\beta : B$, respectively. The equations*

1. $[a]_\alpha \otimes f(\beta|a) = f(a, _)$ and
2. $[b]_\beta \otimes f(\alpha|b) = f(_, b)$,

have solution, and they define Ω -maps $f(\alpha|b) : A \rightarrow \Omega$ and $f(\beta|a) : B \rightarrow \Omega$, given by $f(\beta|a) = [a]_\alpha \Rightarrow f(a, _)$ and $f(\alpha|b) = [b]_\beta \Rightarrow f(_, b)$.

Proof. In a divisible ML-algebra Ω we have $x \otimes (x \Rightarrow y) = x \wedge y$. Since f is faithful $[a]_\alpha = \bigvee_c f(a, c) \geq f(a, c)$, then $[a]_\alpha \geq f(a, _)$. Because $[a]_\alpha \wedge f(a, _) = f(a, _)$ we have

$$[a]_\alpha \otimes ([a]_\alpha \Rightarrow f(a, _)) = [a]_\alpha \wedge f(a, _) = f(a, _).$$

And, we can use the same strategy to proof $f(\alpha|b) = [b]_\beta \Rightarrow f(_, b)$.

We will interpret the Ω -map $f(\beta|a)$ as a classifier in B , defined by relation f , for an entity described by a using the basic monoidal logic Ω .

Applying in the multi-morphism context the principles of Bayes inference: For faithful and total multi-morphisms $f : A \rightarrow B$ and $g : B \rightarrow C$, and Ω -sets $\alpha : A$, $\beta : B$, $\gamma : D$, we have

$$\begin{aligned} [a]_\alpha \otimes (f \otimes g)(\gamma|a)(c) &= (f \otimes g)(a, c) \\ &= \bigvee_b f(a, b) \otimes g(b, c) \\ &= \bigvee_b [a]_\alpha \otimes f(\beta|a)(b) \otimes g(b, c), \end{aligned}$$

then

$$(f \otimes g)(\gamma|a)(c) = [a]_\alpha \Rightarrow ([a]_\alpha \otimes \bigvee_b f(\beta|a)(b) \otimes g(b, c)),$$

i.e.

$$(f \otimes g)(\gamma|a) = \bigvee_b f(\beta|a)(b) \otimes g(b, -),$$

since in a divisible ML-algebra Ω we have $x \Rightarrow (x \otimes y) = x \wedge y$.

When $f : A \rightarrow C$ and $g : B \rightarrow D$ are independent we have

$$(f \otimes g)(\gamma \otimes \delta|a, b)(c, d) = f(\gamma|a)(c) \otimes g(\delta|b)(d).$$

Naturally, if $C = D$ we write $(f \otimes g)(\delta|a, b)(d)$ for $(f \otimes g)(\delta \otimes \delta|a, b)(d, d)$. And in this case we interpret the classifier $(f \otimes g)(\delta|a, b)$ as the combination of two classifiers $f(\delta|a)$ and $g(\delta|b)$, defining entities of $\delta : D$ described by a and b .

Example 3 (Binding) *Drugs are typically small organic molecules that achieve their desired activity by binding to a target site on a receptor. The first step in the discovery of a new drug is usually to identify and isolate the receptor to which it should bind, followed by testing many small molecules for their ability to bind to the target site. This leaves researchers with the task of determining what separates the active (binding) compounds from the inactive (non-binding) ones. Such a determination can then be used in the design of new compounds that not only bind, but also have all the other properties required for a drug (solubility, oral absorption, lack of side effects, appropriate duration of action, toxicity, etc.).*

The DuPont Pharmaceuticals provided a data set to KDD Cup 2001 consisting of 1909 compounds tested for their ability to bind to a target site on thrombin, a key receptor in blood clotting. Of these compounds, 42 were active (bind well) and the others were inactive. Each compound is described by a single feature vector, of observable descriptions, comprised of a class value (A for active, I for inactive) and 139,351 binary features, which describe three-dimensional properties of the molecule. The definitions of the individual bits were not included, they can be seen as not observable descriptions of a compound - we didn't know what each individual bit means, only that they are generated in an internally consistent manner for all 1909 compounds. Biological activity in general, and receptor binding affinity in particular, correlate with various structural and physical properties of small organic molecules. The task proposed on KDD Cup 2001 by DuPont Pharmaceuticals was to determine which of these three-dimensional properties are critical in this case and to learn to accurately predict the class value of a new compound.

Let S be the set of available compounds and suppose what the process of compound classification in laboratory evacuate proposition "Compound a is active" in the a fuzzy logic $\Omega = [0, 1]$. A classification of each compound as active or inactive may be seen as a multi-morphism, in $\text{Set}([0, 1])$,

$$c : S \rightarrow \{A, I\}$$

where $c(a, I)$ and $c(a, A)$ define the truth value of proposition "Compound a is inactive" and "Compound a is active", respectively in $\Omega = [0, 1]$. Each compound in S is described by a set of observable three-dimensional characteristics, measured in laboratory processes, and codified on the dataset. The similarity between compound must be codified by a Ω -set $\alpha : S$. In the case of \bar{x} be an observable three-dimensional structure of a compound in $\alpha : S$ it can be seen as a compound structure generalization. A description x describe a class of compounds $\alpha : S$ and we defined the truth value of proposition "Compounds satisfying description \bar{x} are active" by

$$c(\beta|\bar{x})(A),$$

where the Ω -set $\beta : \{A, I\}$ codify the similarity between the stat of a compound be "active" or "inactive".

In this example the best description \bar{x} for an active compound in S , can be seen as the description that maximizes $c(\beta|\bar{x})(A)$. However from this notion emerges the need of have a way to codify observable descriptions and the existence of a framework for the selection of a best observable description. In the sequel we give a process to describe entities based on a graphic language. Words in this language are used to codifying relations between observable characteristics of entities in a multi-valued logic.

4 Multi-diagrams

A *multi-diagram* in $Set(\Omega)$ is a multi-graph homomorphism $D : \mathcal{G} \rightarrow Set(\Omega)$ defined by mapping the multi-graph vertices to Ω -sets and multi-arrows to multi-morphisms in $Set(\Omega)$.

Formally, if the multi-graph \mathcal{G} is defined using nodes $(v_i)_{i \in L}$ and by a family of multi-arrows (a_{IJ}) , where the multi-arrow a_{IJ} have by source

$$\{v_i : i \in I \subset L\},$$

and by target

$$\{v_j : j \in J \subset L\}.$$

A multi-graph homomorphism $D : \mathcal{G} \rightarrow Set(\Omega)$ transform every node v_i in a Ω -sets $D(v_i)$ and each multi-arrow

$$a_{IJ} : \{v_i : i \in I \subset L\} \multimap \{v_j : j \in J \subset L\},$$

in a multi-morphism

$$D(a_{IJ}) : \prod_{i \in I} D(v_i) \multimap \prod_{j \in J} D(v_j).$$

The usual definition of limit in Set for a diagram can be extended to multi-diagrams in $Set(\Omega)$. For that we must see category Set as the topos $Set(false, true)$, where $false, true$ define a two element chain with the monoidal structure given

by the logic operator "and" and "true". Recall that the limit for a diagram or multi-diagram D is defined as a $\{false, true\}$ -set, denote by $Lim D$, which is a subobject of a cartesian product defined by the diagram vertices (see [17] or [3]). We use these relation on the limit extension to multi-diagrams in $Set(\Omega)$. Since the cartesian product of Ω -sets $(\alpha_i : A_i)$ was defined in 8 as the Ω -set $\otimes_i \alpha_i : \prod_i A_i$ given by

$$[\cdot = \cdot]_{\otimes_i \alpha_i} = \bigotimes_i [\cdot = \cdot]_{\alpha_i},$$

we define

Definition 9 (Limit of a multi-diagram) Let $D : \mathcal{G} \rightarrow Set(\Omega)$ be a multi-diagram where \mathcal{G} have by vertices $(v_i)_{i \in L}$. Its limit $Lim D$ is a subobject of the multi-diagram vertices cartesian product:

$$Lim D \leq \prod_{i \in L} M(v_i)$$

given by

$$Lim D : \prod_{i \in L} M(v_i) \rightarrow \Omega$$

such that

$$(Lim D)(\bar{x}_1, a_i, \bar{x}_2, a_j, \bar{x}_3) = [(\bar{x}_1, a_i, \bar{x}_2, a_j, \bar{x}_3)]_{\prod_{i \in L} M(v_i)} \otimes \bigotimes_{f: v_i \rightarrow v_j \in \mathcal{G}} D(f)(a_i, a_j).$$

We see a limit as the result of applying the pattern used on the definition of each multi-morphism in the cartesian product of its vertices. This definition satisfies the usual universal property when the object classifier used $Set(\Omega)$ is the two-element chain, $2 = \{false, true\}$, with the monoidal structure given by "and" and "true". In other words this definition coincide with the classical one on the context of classical logic.

Note what we can see the limit for a multi-diagram D as a multi-morphism by selecting a set of source Ω -sets and a set of targets. The canonical multi-morphism associated to a multi-diagram D have by source $s(D)$ the union of sources used to define the diagram multi-morphisms and have by target $t(D)$ the union of targets of D multi-arrows.

Example 4 By definition the multi-diagram D given bellow have by limit the Ω -map

$$Lim D : A_0 \times A_1 \times A_2 \times A_3 \times A_4 \times A_5 \rightarrow \Omega$$

given by

$$(Lim D)(a_0, a_1, a_2, a_3, a_4, a_5) = [a_0, a_1, a_2, a_3, a_4, a_5] \otimes f(a_0, a_1, a_3, a_4, a_5) \otimes g(a_1, a_2, a_4, a_5) \otimes h(a_2, a_3).$$

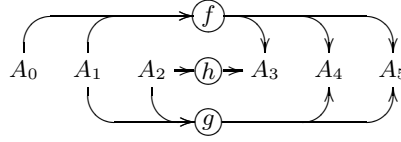


Fig. 4. Multi-diagram.

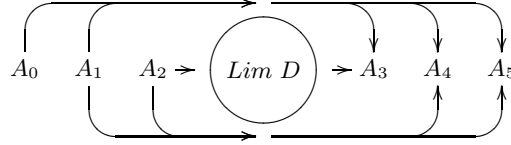


Fig. 5. Multi-diagram limit functionality.

The limit of a multi-diagram collapses the diagram into a multi-morphism by internalizing all the interconnections, thus delivering a multi-diagram as a whole.

In this sense the equalizer of a parallel pair of multi-morphisms $R, S : X \rightarrow Y$ is defined by

$$Lim(R = S) : X \times Y \rightarrow \Omega$$

where

$$Lim(R = S)(x, y) = [x, y] \otimes R(x, y) \otimes S(x, y).$$

And, the pullback of $R : X \rightarrow U$ and $S : Y \rightarrow U$ is the multi-morphism

$$Lim(R \otimes_U S) : X \times U \times Y \rightarrow \Omega$$

where

$$Lim(R \otimes_U S)(x, u, y) = [x, u, y] \otimes R(x, u) \otimes S(y, u).$$

Given a discrete multi-diagram D its limit is denoted by $\Pi_v D(v)$ given by

$$\Pi_v D(v)(\bar{x}) = [\bar{x}]_{\otimes_i D(i)},$$

i.e. when $\bar{x} = (x_1, x_2, \dots, x_n)$, $\Pi_v D(v)(\bar{x}) = [x_1, x_2, \dots, x_n]$.

The presented definition for limit simplifies the proof of:

Proposition 5 (Existence of limit in $Set(\Omega)$) *Every multi-diagram $D : \mathcal{G} \rightarrow Set(\Omega)$ have limit, i.e. exists a multi-morphism $f \leq \prod_{v \in \mathcal{G}} D(v)$ such that $Lim D = f$.*

But more interesting is the fact what we can show the opposite for basic logics:

Proposition 6 *If Ω is a divisible ML-algebra, then for every Ω -map*

$$g : A_0 \times A_1 \times \dots \times A_n \rightarrow \Omega$$

and Ω -map $(\alpha_i : A_i)$ such that

$$g(x_1, x_2, \dots, x_n) \leq [x_1, x_2, \dots, x_n]$$

there is a multi-diagram $D : \mathcal{G} \rightarrow \text{Set}(\Omega)$ such that

$$\text{Lim } D = g$$

We may proof this just by showing that multi-diagram

$$f : A_0 \rightarrow A_1 \times \dots \times A_n$$

where

$$f(x_1, x_2, \dots, x_n) = [x_1, x_2, \dots, x_n] \Rightarrow g(x_1, x_2, \dots, x_n)$$

have by limit g .

We can see a multi-diagram as a way to express dependencies between classes of entities. When the limit of a diagram D is an Ω -object α we want to see the diagram as way to codify α or a model for α . But for this type of relationship be useful we need to have a language to codify the diagram structure. Define this languages is one of the goals for this work.

Let D be a multi-diagram with vertices (v_i) , having by arrow interpretations faithful and total multi-morphisms and let a_i be an observable descriptions of an entity in $D(v_i)$. The limit in $\text{Set}(\Omega)$ of D , where Ω is a divisible ML-algebra, defines the classifier

$$(\text{Lim } D)(D(v_1)|a_2, \dots, a_n)$$

such that

$$[a_2, \dots, a_n] \otimes (\text{Lim } D)(D(v_1)|a_2, \dots, a_n) = \bigotimes_{f: v_i \rightarrow v_j \in \mathcal{G}} [a_i] \otimes D(f)(D(v_j)|a_i)(a_j),$$

i.e.

$$(\text{Lim } D)(D(v_1)|a_2, \dots, a_n) = [a_2, \dots, a_n] \Rightarrow \bigotimes_{f: v_i \rightarrow v_j \in \mathcal{G}} [a_i] \otimes D(f)(D(v_j)|a_i)(a_j),$$

which can be seen as the combination of classifiers related through diagram D to predicted $D(v_1)$. This expression is simplified when in D we don't have multi-arrows with source v_1 , we have

$$(\text{Lim } D)(D(v_1)|a_2, \dots, a_n) = \bigotimes_{f: v_i \rightarrow v_j \in \mathcal{G}} D(f)(D(v_j)|a_i)(a_j).$$

We see the limit of diagram as a generalization for multi-morphism composition of a chain of composable multi-morphisms. This interpretation allows the definition of a semantic for circuits, when we assume a dependence between execution of circuit componentes. This point of view is also used to extend the classic notion of commutative diagram to fuzzy structures. For that we assume that a set of vertices $s(D)$ was selected in a diagram D . The set $s(D)$ is called the set of *sources* for diagram D .

Definition 10 (Commutativity of multi-diagrams) Let D be a multi-diagram where we select $s(D)$ as set of sources. If V is the cartesian product defined by all the vertices of D not in $s(D)$. The multi-diagram D is commutative for $s(D)$ if

$$\bigvee_{\bar{n} \in V} (\text{Lim } D)(\bar{s}, \bar{n}) = \bigvee_{\bar{n} \in V} \left(\prod_i D(i) \right)(\bar{s}, \bar{n}),$$

for every $\bar{s} \in \prod_{i \in s(D)} D(i)$. It is λ -commutative if

$$\left(\bigvee_{\bar{n} \in V} (\text{Lim } D)(\bar{s}, \bar{n}) \Leftrightarrow \bigvee_{\bar{n} \in V} \left(\prod_i D(i) \right)(\bar{s}, \bar{n}) \right) \geq \lambda,$$

for every $\bar{s} \in \prod_{i \in s(D)} D(i)$.

In other words, a multi-diagram is commutative if the multi-morphism defined by its limit, with the selected sources, is total.

Example 5 Lets $\text{Set}([0,1])$ defined by the product logic, \mathbb{R} be the set of real number and \oplus be a relation defined by the multi-morphism $\oplus : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ given by Gaussian function

$$\oplus(x, y, z) = e^{-\frac{(z-x-y)^2}{2}}.$$

The diagram D , presented in fig. 6, with sources $\alpha_0 : \mathbb{R}$ and $\alpha_1 : \mathbb{R}$, where $=$ is

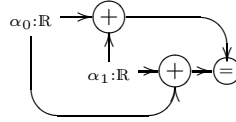


Fig. 6. Multi-diagram D codifying $\alpha_0 + \alpha_1 = \alpha_1 + \alpha_0$.

defined as equality in \mathbb{R} , is commutative for every $x_0, x_1 \in \mathbb{R}$, when we have by densities in α_0 and α_1 ,

$$\alpha_0(x, y) = e^{-\frac{(x-x_0)^2}{2} - \frac{(y-x_0)^2}{2}} \text{ and } \alpha_1(x, y) = e^{-\frac{(x-x_1)^2}{2} - \frac{(y-x_1)^2}{2}}.$$

Because, using the definition presented to the multi-diagram limit, we have

$$\begin{aligned} (\text{Lim } D)(x, y, w) &= \oplus(x, y, w) \otimes \oplus(y, x, w) \otimes [x, y, w] \\ &= \oplus(x, y, w) \otimes \oplus(y, x, w) \otimes [x] \otimes [y] \otimes [w] \\ &= e^{-\frac{(w-x-y)^2}{2}} \cdot e^{-\frac{(w-y-x)^2}{2}} \cdot e^{-\frac{(x-x_0)^2}{2} - \frac{(x-x_0)^2}{2}} \cdot e^{-\frac{(y-x_1)^2}{2} - \frac{(y-x_1)^2}{2}} \cdot 1 \\ &= e^{-(w-x-y)^2 - (x-x_0)^2 - (y-x_1)^2} \end{aligned}$$

then, since $e^{-(w-x-y)^2} \leq 1$, we have $e^{-(w-x-y)^2-(x-x_0)^2-(y-x_1)^2} \leq e^{-(x-x_0)^2-(y-x_1)^2}$, and

$$\begin{aligned} \bigvee_w (\text{Lim } D)(x, y, w) &= e^{-(x-x_0)^2-(y-x_1)^2} \\ &= [x]_{\alpha_0} \otimes [y]_{\alpha_1} \\ &= \bigvee_w \alpha_0(x, x) \otimes \alpha_1(y, y) \otimes (w, w). \end{aligned}$$

This proves the commutativity for diagram D when its sources have the fixed distributions. The diagram D' , presented on fig. 7, having by source the $[0, 1]$ -set

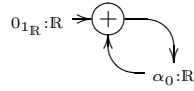


Fig. 7. Multi-diagram D' codifying $0_{1R} + \alpha_0 = \alpha_0$.

$\alpha_0 : \mathbb{R}$, defined by distribution

$$\alpha_0(x, y) = e^{-\frac{(x-x_0)^2}{2} - \frac{(y-x_0)^2}{2}},$$

is also commutate, when the $[0, 1]$ -set $0_R : \mathbb{R}$ is defined by distribution

$$0_R(x, y) = \lambda \cdot e^{-\frac{x^2}{2} - \frac{y^2}{2}},$$

where the parameter λ is a truth value selected in $[0, 1[$. Since

$$\begin{aligned} (\text{Lim } D')(x, y, x) &= \oplus(x, y, x) \otimes [x, y, x] \\ &= \oplus(x, y, x) \otimes [x] \otimes [y] \otimes [x] \\ &= e^{-\frac{(x-x-y)^2}{2}} \cdot e^{-\frac{(x-x_0)^2}{2} - \frac{(x-x_0)^2}{2}} \cdot \lambda \cdot e^{-\frac{y^2}{2} - \frac{y^2}{2}} \cdot e^{-\frac{(x-x_0)^2}{2} - \frac{(x-x_0)^2}{2}} \\ &= \lambda \cdot e^{-(x-x-y)^2 - 2(x-x_0)^2 - y^2} \end{aligned}$$

and

$$\begin{aligned} \bigvee_y (\text{Lim } D')(x, y, x) &= \lambda \cdot e^{-(x-x-0)^2 - 2(x-x_0)^2 - 0^2} \\ &= \lambda \cdot e^{-2(x-x_0)^2} \\ &= \lambda \otimes [x] \otimes [x] \\ &= \bigvee_y \alpha_0(x, x) \otimes 0_R(y, y) \otimes \alpha_0(x, x). \end{aligned}$$

However, if we change in diagram D' the interpretation of \oplus using the new distribution

$$\oplus(x, y, z) = \lambda' \cdot e^{-\frac{(z-x-y)^2}{2}},$$

depending from a parameter $\lambda' \in [0, 1[$. We have

$$(\text{Lim } D')(x, y, x) = \lambda' \cdot \lambda \cdot e^{-(x-x-y)^2 - 2(x-x_0)^2 - y^2},$$

thus

$$\bigvee_v (\text{Lim } D')(x, y, x) = \lambda' \cdot \bigvee_y \alpha_0(x, x) \otimes 0_R(y, y) \otimes \alpha_0(x, x).$$

Then, since we are working in a multiplicative logic, we have

$$\left(\bigvee_v (\text{Lim } D')(x, y, x) \Leftrightarrow \bigvee_v (\alpha_0(x, x) \otimes 0_{\mathbf{R}}(y, y) \otimes \alpha_0(x, x)) \right) \geq \lambda',$$

which means that D' is λ' -commutative.

Naturally, if a diagram is λ -commutative, it also is λ' -commutative, when $\lambda' < \lambda$. When for every $\lambda > \perp$ the diagram isn't λ -commutative it is called a non-commutative diagram.

However, when we see a multi-diagram as a way of specify a architectural connectors, in the sense of [24], we may want to interpret diagrams by collapsing the joint execution of its components, generalizing the notion of parallel composition. We may do this through the symmetry between operators \otimes and \vee on classic logic. We define:

Definition 11 (Colimit of multi-diagrams) *Given a multi-diagram D in $\text{Set}(\Omega)$ with vertices (v_i) the colimit is defined by the multi-morphism*

$$\text{coLim } D \leq \prod_i M(v_i)$$

i.e.

$$\text{coLim } D : \prod_i M(v_i) \rightarrow \Omega$$

given by

$$(\text{coLim } D)(\bar{x}_1, a_i, \bar{x}_2, a_j, \bar{x}_3) = [\bar{x}_1, a_i, \bar{x}_2, a_j, \bar{x}_3]_{\prod_i M(v_i)} \otimes \bigvee_{f: v_i \rightarrow v_j \in \mathcal{G}} D(f)(a_i, a_j).$$

This definition allows the formalization of knowledge integration. Colimits capture a generalized notion of parallel composition of components in which the designer makes explicit what interconnections are used between components. We can see this operation as a generalization of the notion of superimposition as defined in [4].

The colimit for a multi-diagram D can be used to define multi-morphism by selection of a set of source and a set of targets. The canonical multi-morphism associated to a multi-diagram D , using colimit, have by source $s(D)$ the union of sources used to define the diagram multi-morphisms and have by target $t(D)$ the union of targets of D multi-morphisms.

Example 6 *By definition the multi-diagram D , presented in fig. 4, have by colimit the Ω -map*

$$\text{coLim } D : A_0 \times A_1 \times A_2 \times A_3 \times A_4 \times A_5 \rightarrow \Omega$$

given by

$$(\text{coLim } D)(a_0, a_1, a_2, a_3, a_4, a_5) = [a_0, a_1, a_2, a_3, a_4, a_5] \otimes (f(a_0, a_1, a_3, a_4, a_5) \vee g(a_1, a_2, a_4, a_5) \vee h(a_2, a_3)).$$

In this sense the coequalizer of a parallel pair of multi-morphisms $R, S : X \rightharpoonup Y$ is defined by the multi-morphism $colim(R = S) : X \times Y \rightarrow \Omega$ given by

$$coLim(R = S)(x, y) = [x, y]_{X \times Y} \otimes (R(x, y) \vee S(x, y)).$$

And the pushout of $R : X \rightharpoonup U$ and $S : Y \rightharpoonup U$ is the multi-morphism

$$coLim(R \oplus_U S) : X \times U \times Y \rightarrow \Omega$$

given by

$$coLim(R \oplus_U S)(x, u, y) = [x, y]_{X \times Y} \otimes (R(x, u) \vee S(y, u)).$$

When D is a discrete diagram colimit coincide with the limit of D , and in this case, we write

$$\coprod_v D(v) = \prod_v D(v).$$

Naturally

Proposition 7 (Existence of coLimit in $Set(\Omega)$) *Every multi-diagram $D : \mathcal{G} \rightarrow Set(\Omega)$ have colimit.*

Since $Set(\Omega)$ have limit and colimit of multi-diagrams we use it, in the following, as "Universe of Discourse" to construct model for structures specified by diagrams on the monoidal logic described in Ω .

Example 7 (Genome) *The genomes of several organisms have now been completely sequenced, including the human genome. Interest within bioinformatics is therefore shifting somewhat away from sequencing, to learning about the genes encoded in the sequence. Genes code for proteins, and these proteins tend to localize in various parts of cells and interact with one another, in order to perform crucial functions. A data set presented to KDD Cup 2001 consists of a variety of details about the various genes of one particular type of organism. The two tasks proposed for the Data Analysis Challenge were to predict the functions and localizations of the proteins encoded by the genes. A gene/protein can have more than one function, and more than one localization. The other information from which function and localization can be predicted includes the class of the gene/protein, the phenotype (observable characteristics) of individuals with a mutation in the gene (and hence in the protein), and the other proteins with which each protein is known to interact. The dependencies associated to the problem may be expressed by the multi-diagram D presented by fig. 8. The diagram limit defines a morphism which characterize the involved entities:*

$Gene \times Class \times Phenotype \times Gene \times Interaction.type \times Function \times Localization$

$$\downarrow \\ \Omega$$

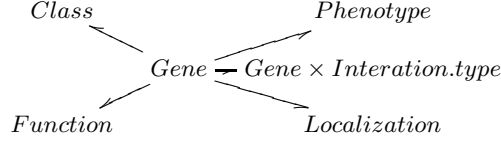


Fig. 8. Dependencies between attributes.

This map can be seen as a data set where we can compute $\alpha : Function \times Localization$ describing the similarity between of pair on $Functions \times Localization$. Given a description \bar{x} for a gene, the Ω -map

$$(\text{Lim } D)(\alpha|\bar{x})(f, l)$$

reflect the truth value in Ω of the proposition "the class of genes characterized by x have function f and localization l ".

In this sense a multi-diagram can be aggregate in a relation via its limit. In the following sections we will describe the inverse problem: Define a multi-diagram having by limit an "approximation" to a given multi-morphism. By this we mean, the possibility of express graphically a fuzzy relations between attributes aggregate in a multi-morphism codifying a data set.

5 Specifying libraries of components

In Computer Sciences a formal grammar is an abstract structure that describes a formal language. Formal grammars are classified into two main categories: generative or analytic.

The generative grammars are the most well-known kind. It is a set of rules by which all possible strings in the language to be described can be generated by successively rewriting strings from a designated start symbol. An analytic grammar, in contrast, is a set of rules that assume an arbitrary string to be given as input, and which successively reduces or analyzes the input string yield a final boolean, "yes/no", result indicating whether or not the input string is a member of the language described by the grammar.

The languages used in this work are expressed through a type of generative grammar where words are configurations defined using components selected from a library. Each component have associated a set of requisites and a configuration is valid in the language if every requisite for the used components are satisfied. The use of this type of structure and the definition of its semantic was motivated on the Architectural Connectors domain which emerged as a powerful tool for supporting the description of the overall organization of systems in terms of components and their interactions [12] [15] [6]. According to [23], an architectural connector can be defined by a set of roles and a glue specification. The roles of a connector type can be instantiated with specific components of the

system under construction, which leads to an overall system structure consisting of components and connector instances establishing the interactions between the components.

Let $Chains$ be the forgetful functor from the category of total ordered sets and its homomorphisms to Set , the category of all sets. If we interpret a set Σ as a set of symbols or signs, objects on the comma category $(Chains \downarrow \Sigma)$ can be seen as words defined by strings over *alphabet* Σ . A set of signs Σ equipped with a partial order \leq is called a *ontology*. Given signs λ_0 and λ_1 on an ontology (Σ, \leq) such that $\lambda_0 \leq \lambda_1$, λ_1 is called a *generalization* of λ_0 and λ_0 is called a *particularization* of λ_1 .

Given $w \in (Chains \downarrow \Sigma)$, we write $w : |w| \rightarrow \Sigma$, where $|w|$ denotes the chain used on the indexation w , and it is interpreted as an ordered sequence of symbols from Σ .

An ontology (Σ, \leq) is called a *bipolarized ontology*, if we have a nilpotent operator $(\cdot)^+ : \Sigma \rightarrow \Sigma$, such that $\Sigma = \Sigma_I \cup \Sigma_O$, where $(\Sigma_I)^+ = \Sigma_O$, and preserving the ontology structure, i.e. given signs λ_0 and λ_1 if $\lambda_0 \leq \lambda_1$ then $\lambda_0^+ \leq \lambda_1^+$. Set Σ_I is called the set of Σ *input symbols* and Σ_O is called the set of Σ *output symbols*. If the symbol λ is an input symbol, λ^+ is called the dual of λ and it is an output symbol. A bipolarized ontology will be denoted by (Σ^+, \leq) .

Using lifting we define for every word $w \in (Chains \downarrow \Sigma^+)$, the words:

1. $o(w) \in (Chains \downarrow \Sigma_O)$ defined by all output symbols in w and
2. $i(w) \in (Chains \downarrow \Sigma_I)$ defined by all input symbols in w .

$$\begin{array}{ccc}
 |i(w)| & \xrightarrow{i(w)} & \Sigma_I \\
 \downarrow \subseteq & \lrcorner & \downarrow \subseteq \\
 |w| & \xrightarrow{w} & \Sigma
 \end{array}
 \qquad
 \begin{array}{ccc}
 |o(w)| & \xrightarrow{o(w)} & \Sigma_O \\
 \downarrow \subseteq & \lrcorner & \downarrow \subseteq \\
 |w| & \xrightarrow{w} & \Sigma
 \end{array}$$

Fig. 9. Pullbacks used to select input and output signs from word w .

Given a word w , we define $\Sigma(w)$ as the set of symbols used in w , $\Sigma(w) \subset \Sigma^+$. On an bipolarized ontology, let w and w' be two words, $w \otimes w'$ is a substring from concatenation $w.w'$ inductively described by the following algorithm:

Algorithm 1 (*Input:* $w, w' \in \Sigma^+$ *Output:* w_i, w'_i)

1. let $w_0 = w$ and $w'_0 = w'$.
2. let λ be the first output symbol, in w_i having its dual λ^+ in w'_i or one of its generalizations :
 - (a) w_{i+1} is generated removing the first occurrence of λ from w_i ;
 - (b) w'_{i+1} is generated removing the first occurrence of λ^+ or a λ^+ generalization from w'_i

3. the step 2 is repeated while there are signs in $\Sigma(w_{i+1})$ with dual or generalization in $\Sigma(w_{i+1})$.

In this sense we can see the word $w \otimes w'$ as the result of the ordered elimination of output symbols on w and input symbols on w' linked by duality.

From the definition of operator \otimes we can proof:

Proposition 8 *For every pair of words w and w' in a bipolarized ontology (Σ, \leq) we have:*

1. $w \otimes (w' \otimes w'') = (w \otimes w') \otimes w''$;
2. $w' \otimes w = w \otimes w'$, if w' (and w) not have the dual neither one of its generalizations of signs from w (and w' , respectively);
3. $w \otimes \perp = \perp \otimes w = w$.

For our goal of finding a framework for library specification, we supposed processes inputs and outputs requirements codified over signs from a polarized ontology (Σ^+, \leq) . Thus *the universe of libraries* having components requirements codified over the polarized ontology Σ^+ can be seen as the comma category

$$(\text{Chains} \downarrow (\text{Chains} \downarrow \Sigma^+)).$$

With this we mean that a library is a list for componentes specified using words defined over Σ^+ .

A *library specification* is a map $L : |L| \rightarrow (\text{Chains} \downarrow \Sigma^+)$, where each node in the chain $|L|$ is called a *component label* or a *sign* in the library. Given a component label $r \in |L|$ we can see $L(r) = w : |w| \rightarrow \Sigma^+$ as the specification of the component input requirements, $i(w)$, and its output requirements $o(w)$. In this sense we see a library as an oriented multi-graph having by multi-arrows a selection of objects in $(\text{Chains} \downarrow \Sigma^+)$ and having by nodes objects from $(\text{Chains} \downarrow \Sigma_I)$.

Let $L \in (\text{Chains} \downarrow (\text{Chains} \downarrow \Sigma^+))$, if $L(r) = w$, r is interpreted as a dependence between families of nodes $i(w)$ and $o^+(w)$. And in this case we write

$$r : i(w) \rightarrow o^+(w),$$

or for short $r \in L$, defining a multi-arrow in the multi-graph $\mathcal{G}(L)$ associated to the library L .

A homomorphism between libraries is a morphism in $(\text{Chains} \downarrow (\text{Chains} \downarrow \Sigma^+))$. Every morphism $f : L_0 \rightarrow L_1$ between libraries L_0 and L_1 have associated a multi-graph homomorphism $\mathcal{G}(f) : \mathcal{G}(L_0) \rightarrow \mathcal{G}(L_1)$, defining a correspondence between signs and a correspondence between component labels in L_0 and L_1 , preserving component requirements.

Naturally, we may define an order relation between libraries, we write $L_0 \leq L_1$ if for every $r \in |L_0|$ we have $r \in |L_1|$, i.e. every componente existent in L_0 is in L_1 . In this case L_0 is called a *sublibrary* of L_1 and the associated homomorphism $f : L_0 \rightarrow L_1$ is called the *library inclusion*.

We denote by L^* the *free monoid* on L for operator \otimes . Formally, given a library $L \in (\text{Chains} \downarrow (\text{Chains} \downarrow \Sigma^+))$ we define L^* as the \otimes -closure of L , i.e. it is the least library in $(\text{Chains} \downarrow (\text{Chains} \downarrow \Sigma^+))$ such that:

1. every word generated using signs of L is a label in L^* ;
2. the empty word defines a label for a component having empty requisites
 $\perp : \perp \rightarrow \perp$;
3. L is a sublibrary of L^* ;
4. if $s \in L^*$ such that $s = r_1 \otimes r_2 \otimes \dots \otimes r_n$ then

$$L^*(s) = L(r_1) \otimes L(r_2) \otimes \dots \otimes L(r_n).$$

Note what the empty word \perp is a label in L^* . Since L is a sublibrary of L^* the requirements of a label in L^* can be interpreted as the requirements for the plugging of the components used on the label definition. In this sense a word in L^* can be seen as a circuit defined by the plugging of components from L .

A library is a formal system where we may stratify in different levels of abstraction. The level of abstraction of a circuit is define by the number of steps of refinement need to obtain an equivalent circuit using only atomic components. In order to presente what we mean by a circuit refinement, note that in an ontology (Σ^+, \leq) the order defined for sign can be lifted to words. We write $\lambda_0 \dots \lambda_n \leq \lambda'_0 \dots \lambda'_n$ if and only if $\lambda_i \leq \lambda'_i$ in (Σ^+, \leq) . When for two words from ontology we have $w \leq w'$, w' is called a *generalization* of w on the ontology.

Circuit refinement is based on the notion of *semantic for a library* in L , and it is a pair of equivalence relations (\equiv_l, \equiv_w) , where \equiv_l is defined for labels in L^* and \equiv_w is defined for words in Σ^* , such that:

$$\text{If } l_0 \equiv_l l_1 \text{ then } L^*(l_0) \equiv_w L^*(l_1).$$

In $(L^*, \equiv_l, \equiv_w)$ a label s is called a *decomposable componente* if there are words s_0 and s_1 such that:

$$s \equiv_l s_0 \otimes s_1$$

We called to a labels that can't be decomposable an *atomic componente*. In this sense if a label in L^* is atomic, it is a label in library L .

A *normal form* presentation for a label $s \in L^*$ is a sequence of atomic components

$$(r_0, r_1, r_2, \dots, r_n)$$

such that

$$s \equiv_l r_0 \otimes r_1 \otimes r_2 \otimes \dots \otimes r_n.$$

Given a library L we call *library of atomic components* of $(L^*, \equiv_l, \equiv_w)$ to the library

$$L_{at} \leq L$$

such that, $s \in L_{at}$ if and only if s is atomic in $(L^*, \equiv_l, \equiv_w)$.

We want to describe structures, like Architectural connectors, using a graphic language to describe the global organization of complex structures having by resource simplest ones. For that, each component is associated to a graphic presentation and the circuit specification result of the linkage between components satisfying a set of rules and a glue specification [23]. The essence of our approach is to provide a general framework that gives circuit explicit semantic status. To

formalize this, for a library L we associated a multi-graph $\mathcal{G}(L)$, having by nodes symbols from Σ_I , and by multi-arcs componentes such that each component s have by input $i(L(s))$ and output $o(L(s))$. By $\mathcal{G}^*(L)$ we denote the comma category

$$(Mgraph \downarrow \mathcal{G}(L))$$

having by objects homomorphisms defined between a multi-graph and the multi-graph $\mathcal{G}(L)$.

Trivially, for a library L with polarized ontology (Σ^+, \leq) , any diagram $D \in \mathcal{G}^*(L)$ can be codified as a library

$$L(D) \in (Chains \downarrow (Chains \downarrow \Sigma^+)),$$

having as component labels multi-arcs, from D , each one must be associated to a word defined concatenating, in a single word, the multi-arc sources vertices labels and its targets vertices dual labels. Given a diagram $D \in \mathcal{G}^*(L)$ defined through:

1. the source map $i : |D| \rightarrow (Chains \downarrow \Sigma_I)$ and
2. the target map $o : |D| \rightarrow (Chains \downarrow \Sigma_I)$.

Any library $L(D)$ have associated two words, defined using symbols from Σ_I ; This words are its input requisites $i(D)$ and its output structures $o(D)$, where:

1. $i(D)$ is the word define concatenating labels belonging to vertices without input multi-arc;
2. $o(D)$ is a word defined by concatenation of the dual of labels belonging to vertices without output multi-arc.

On the category $\mathcal{G}^*(L)$, for every pair of diagrams D and D' , we define the diagram $D \otimes D'$ by gluing together vertices with equal labels belonging to $o(D)$ and $i(D')$, taken others as distinct. The order used to gluing vertices must respect the order given by the chain of symbols used to define the words $o(D)$ and $i(D')$.

When \otimes is restricted to pairs of diagrams D and D' such that $i(D) = o(D')$, we used this operator as a "composition" between relations specified using multi-graphs. With it we define a category having by objects words from $(Chains \downarrow \Sigma_I)$ and by morphisms diagrams from $\mathcal{G}^*(L)$. Given a diagram D , the fact of $i(D) = w$ and $o(D) = w'$ is denoted by $D : w \rightharpoonup w'$. Given diagrams D and D' from $\mathcal{G}^*(L)$, if D' is a subobject of D , denoted by writing $D' \leq D$, if here is an epimorphism in $\mathcal{G}^*(L)$ from D' to D , or equivalently, if there is a decomposition $D = D'' \otimes D' \otimes D'''$.

A diagram D is *decomposable* if there are two not null subobjects D' and D'' such that $D = D' \otimes D''$. If a diagram isn't decomposable it is called *atomic*. Let $\mathcal{G}_{at}^*(L)$ be the class of atomic diagrams in $\mathcal{G}^*(L)$. We can see atomic diagrams as building blocks for generate diagrams. A functor $F : \mathcal{G}^*(L) \rightarrow \mathcal{G}^*(L)$, where L have a semantic, is called a *diagram refinement* if:

1. $F(D \otimes D') \equiv_l F(D) \otimes F(D')$, i.e. the refinement of a diagram is semantically equivalente to the refinement for its parts;

2. $F(D) \equiv_l D$, i.e. the refinement of a diagram is semantically equivalent to it self;
3. $F(D) = D$ if and only if $D \in \mathcal{G}_{at}^*(L)$, i.e. atomic elements cannot be simplified.

A refinement can be seen as a rewriting rule allowing unpacking subdiagram encapsulations. If a diagram is a fixed-point for the refinement function we say it is in *normal form*. And since diagrams are finite structure, for every diagram D we can find, at least, a representation of D in normal form in a finite number of steps.

A diagram refinement $F : \mathcal{G}^*(L) \rightarrow \mathcal{G}^*(L)$ have the nice property of defining a partial order in $\mathcal{G}^*(L)$, denoted by \leq_F and where $D \leq_F D'$ is true if $F(D') = D$, i.e. if D is a refinement of D' through F , and in this case we call to D' a *generalization* of D .

Example 8 (Signatures as libraries) *A signature can be expressed through a library of components, where each component represents a function symbol where its arity is codified on the component requirements. Formalizing this: Following [16] a signature $\Sigma = (S, T, ar)$, with type symbols from S , consists of a finite set T of function symbols (or operators) f, g, \dots where each function f has an arity $ar(f) = (< a_i >_I, b)$ defined by a chain of input type symbols and one output type symbol. In this case we write $i(f) = < a_i >_I$ and $o(f) = < b >$. We can sort the set T of symbols function and taking these symbols as labels of components having its requirements codified over the polarized alphabet S^+ generated from S . The set S^+ is defined adding a new dual symbol a^+ for each type symbol a in S . The library associated to the signature Σ , will be denoted by $L(\Sigma)$.*

A constant, of type a , is a function symbols in a signature with arity $(<>, a)$, i.e. without input and having a as output. It is usual to take a countable infinite set of variables for each type used on the signature. They are codified on a diagram using inputs on components without associated links, and defining the set of diagram sources.

Bellow we present some examples of libraries associated to models generated by machine learning algoritmos in [27] and used on the following for the presentation of examples by describing fuzzy structures :

Example 9 (Binary Library $\mathbf{L}_B(S, C)$) *Binary libraries are define using a set of component labels C and a set of type signs S . A binary library $L_B(S, C)$ presuppose the existence of a sign $l \in S$, interpretable as the set Ω of truth values in $Set(\Omega)$, and a constant $\top : l^+$ in C interpreted as true. In S we must have defined components of type $=_s : ssl^+$, one for each symbol $s \in S$, interpreted as a similarity $[\cdot = \cdot]$ on the interpretation for s , and also components of type $c : s^+$, where $c \in C$ and $s \in S$, interpreted as a constants selected on the interpretation for s .*

Since data sets or tables can be codified using this primitives, binary libraries are also called data set libraries

Example 10 (Linear Library $\mathbf{L}_L(S, C)$) Linear libraries $L_L(S, C)$, extend binary libraries, are defined using similarities $=_s: ssl^+$, components specified as $\geq_s: ssl^+$, for each symbol $s \in S$, interpretable as a total order and constant components $c: s^+$, where $c \in C$ and $s \in S$.

Linear library are associated to processes for the discretization of continuous domains in this sense they are also called grid libraries.

Example 11 (Additive Library $\mathbf{L}_A(S, C)$) An additive libraries $L_A(S, C)$ is an extension to a linear libraries. They are defined using equality and order components $=_s: ssl^+$, $\geq_s: ssl^+$, with a component specified as $+_s: sss^+$, interpretable as an addition for all symbol $s \in S$, and constantes $b: s^+$ where $b \in C$ and $s \in S$.

Example 12 (Multiplicative Library $\mathbf{L}_M(S, C)$) Multiplicative libraries $L_M(S, C)$ are extensions to additive libraries. They are defined using components $=_s: ssl^+$, $\geq_s: ssl^+$, $+_s: sss^+$, also have a component $\times_s: sss^+$, for each symbol $s \in S$, interpretable as a multiplication and constantes $b: s^+$ for $b \in C$ and $s \in S$.

6 Modeling libraries and Graphic Languages

Since libraries and multi-graphs have structural compatibility it is natural to assume the soundness for library semantic in $Set(\Omega)$ as equivalente to the library structural preservation.

A model for a library $(L^*, \equiv_l, \equiv_w)$ in $Set(\Omega)$ is a multi-graph homomorphism M from the library parser graph $\mathcal{G}(L^*)$ to $Set(\Omega)$,

$$M : \mathcal{G}(L^*) \rightarrow Set(\Omega)$$

such that

1. equivalente componentes are interpreted as the same multi-morphism, i.e.
 $M(r) = M(r')$ if $r \equiv_l r'$;
2. transform componente gluing in multi-morphism composition, i.e.

$$M(r \otimes r') = M(r) \otimes M(r');$$

3. preserves componente requirements and truth value distribution, i.e. if $r : w \rightharpoonup w'$ then

$$M(r) : M(w) \rightharpoonup M(w') \text{ and } M(r)^\circ \otimes M(w) \otimes M(r) = M(w');$$

4. preserves sign ontological structure, i.e. if sign l is a generalization for sign l' (i.e. if $l' \leq l$) then $M(l') \leq M(l)$;
5. words are mapped to as chains of Ω -set products defined in 8, i.e. if $w = s_1 s_2 \dots s_n$ then $M(w) = \Pi_i M(s_i)$;
6. equivalente words are mapped to the same Ω -set defined in 7, i.e. if $w \equiv_w w'$ then $M(w) = M(w')$.

In other words a model transform multi-arcs into multi-morphisms preserving its structure and the semantic induced through relations \equiv_l and \equiv_w . Property (3) imposes the preservation of truth values distribution by componente interpretation. The class of models for a library $(L^*, \equiv_l, \equiv_w)$ is used, in the sequel, on definition of a category of models

$$Mod(L^*, \equiv_l, \equiv_w).$$

A model for L^* can be defined lifting interpretation of atomic components to circuits. For that we must note that, since a model preserves componente gluing, it can be defined by fixing interpretations for its atomic components. This is expressed by the following completion principle:

Proposition 9 (Universal property) *Let L_{at} be the sublibrary defined by atomic components in $(L^*, \equiv_l, \equiv_w)$. Every multi-graph homomorphism*

$$M : \mathcal{G}(L_{at}) \rightarrow Set(\Omega),$$

defines a unique model

$$M^* : \mathcal{G}(L^*) \rightarrow Set(\Omega),$$

for $(L^, \equiv_l, \equiv_w)$, such that*

$$M^* \circ i = M,$$

where i is the homomorphism defined by library inclusion.

The proof to this result is made defining $M^*(r) = \otimes_i M(r_i)$ if the circuit r have a normal form given by a sequence $(r_0, r_1, r_2, \dots, r_n)$, i.e. the r_i 's are atomic and

$$r \equiv_l r_0 \otimes r_1 \otimes r_2 \otimes \dots \otimes r_n.$$

For every component label $r \in L$ we called *a realization* for r through M in $Mod(L^*, \equiv_l, \equiv_w)$ or a *Chu representation* of r to the a *epi multi-morphism*

$$M(r) : M(i(r)) \multimap M(o(r)) \text{ such that } M(r)^\circ \otimes M(i(r)) \otimes M(r) = M(o(r)).$$

In this case, if $r = r' \otimes r''$ in L^* , then $M(r)$ can be decomposed in $Set(\Omega)$ as

$$M(r') \otimes M(r'').$$

Since library refinement preserves semantics, it is idempotent with regard to library models, given a model $M \in Mod(L^*, \equiv_l, \equiv_w)$ and if $F : L^* \rightarrow L^*$ is a refinement in library $(L^*, \equiv_l, \equiv_w)$, we have

$$M(F^n(D)) = M(D), \text{ for every configuration } D \in \mathcal{G}(L^*).$$

This property can be used to characterize refinement:

Proposition 10 *A library homomorphism $F : L^* \rightarrow L^*$ having by fixed-points atomic components is a refinement in $(L^*, \equiv_l, \equiv_w)$ if and only if for every model $M \in \text{Mod}(L^*, \equiv_l, \equiv_w)$ we have*

$$M \circ F^n = M$$

for each natural n .

In this sense, for every component r which is a refinement for r' by F , i.e. $r \leq_F r'$, we have, for every library model M , $M(r') = M(r)$.

A library can be seen as an analytic grammar and we can use them to characterize languages. We define the *graphic language* associated to a library L as the set of valid finite configurations using components indexed by L . A configuration of components D is *valid or allowed* in L if

$$D \in \mathcal{G}^*(L),$$

i.e. if D is a multi-graph homomorphism

$$D : \mathcal{G} \rightarrow \mathcal{G}(L).$$

Formally, given a library $L \in (\text{Chains} \downarrow (\text{Chains} \downarrow \Sigma^+))$, a graphic word D defined by L is a finite configuration

$$D \in (Mgraph \downarrow \mathcal{G}(L)).$$

In this sense a word in the language is a multi-graph homomorphisms where the multi-arrows are library components. Since the homomorphism D have $\mathcal{G}(L)$ as codomain it satisfy library constraints and it can be seen, and is called, the *parsing* of word D .

The *graphic language defined by library L* , is denoted by $\text{Lang}(L)$, and it is the comma category $\mathcal{G}^*(L) = (Mgraph \downarrow \mathcal{G}(L))$ of allowed configurations in L . Given an allowed configuration $D : \mathcal{G} \rightarrow \mathcal{G}(L)$ we call \mathcal{G} the *configuration shape* and to $D(\mathcal{G})$ a *word or diagram* on the language defined by L .

Given a configuration $D \in \text{Lang}(L)$ and a model $M \in \text{Mod}(L^*, \equiv_l, \equiv_w)$, we define

1. $i(M(D)) = M(i(D))$ and
2. $o(M(D)) = M(o(D))$,

where $M(i(D))$ and $M(o(D))$ denote Ω -sets in $\text{Set}(\Omega)$ used to give meaning to multi-diagram input and output vertices.

We may collapse the structure of a word interpretation on a multi-morphism using limits.

Definition 12 (Limits as multi-morphisms) *Given a model $M \in \text{Mod}(L^*, \equiv_l, \equiv_w)$, the interpretation for a configuration $D : \mathcal{G} \rightarrow \mathcal{G}(L)$ through M is $\text{Lim } MD$ a multi-morphism*

$$M(i(D)) \multimap M(o(D)).$$

In this case we write $M(D)$ to denote the multi-morphism $\text{Lim } MD$.

Naturally we define

Definition 13 (coLimits as multi-morphisms) *Given a model $M \in Mod(L^*, \equiv_l, \equiv_w)$ and a diagram $D : \mathcal{G} \rightarrow \mathcal{G}(L)$. Its colimit $coLim MD$ can be seen as a multi-morphism*

$$coLim MD : M(i(D)) \multimap M(o(D)).$$

By definition the model for a library preserves component decomposition, which can be extended to multi-diagrams when interpreted in a basic logic.

Proposition 11 *Let Ω be a basic logic. If multi-diagram D is a word on the language defined by library L and if it can be obtain by gluing diagrams D_1 and D_2 , i.e. $D = D_1 \otimes D_2$. An interpretation for word D is the result of composing the interpretation of D_1 and D_2 , i.e.*

$$M(D) = [\cdot = \cdot]_{\otimes H} \Rightarrow (M(D_1) \otimes M(D_2)),$$

where $H = M(i(D_2) \cap o(D_1))$ is the set of Ω -sets that are sources for diagram D_2 and targets for D_1 .

Note that, if $M(D_1)^\circ \otimes \alpha \otimes M(D_1) = \beta$ and $M(D_2)^\circ \otimes \beta \otimes M(D_2) = \gamma$ then $M(D_1 \otimes D_2)^\circ \otimes \alpha \otimes M(D_1 \otimes D_2) = M(D_2)^\circ \otimes M(D_1)^\circ \otimes \alpha \otimes M(D_1) \otimes M(D_2) = \gamma$.

In a basic logic Ω , this strategy can be extended to configurations colimit:

$$colim M(D_1 \otimes D_2) = [\cdot = \cdot]_{\otimes H} \Rightarrow (colim M(D_1) \otimes colim M(D_2)),$$

where $H = M(i(D_2) \cap o(D_1))$.

7 Library descriptive power

Lets define now the structure for the category of models for a library, $Mod(L^*, \equiv_l, \equiv_w)$. It has by objects models

$$M : \mathcal{G}(L^*) \rightarrow Set(\Omega),$$

and by morphisms natural transformations. In this context, taking $D = \mathcal{G}(L^*)$ the graphic library structure, a natural transformation from model M_1 to model M_2 is a pair of epi multi-morphisms $(f, g) : M_1 \Rightarrow M_2$, such that

$$f \otimes M_2(D) = M_1(D) \otimes g.$$

Naturally, by definition of epi multi-morphism,

$$f^\circ \otimes M_1(i(D)) \otimes f = M_2(i(D)) \text{ and } g^\circ \otimes M_1(o(D)) \otimes g = M_2(o(D)).$$

$$\begin{array}{ccc}
M_1(i(D)) & \xrightarrow{f} & M_2(i(D)) \\
M_1(D) \downarrow & & \downarrow M_2(D) \\
M_1(o(D)) & \xrightarrow{g} & M_2(o(D))
\end{array}$$

Fig. 10. Natural transformation (f, g) .

We use the composition for multi-morphism to define the composition of natural transformation. Given two natural transformations $(f_1, g_1) : M_1 \Rightarrow M_2$ and $(f_2, g_2) : M_2 \Rightarrow M_3$ we define

$$(f_1, g_1) \otimes (f_2, g_2) = (f_1 \otimes f_2, g_1 \otimes g_2).$$

A model M have by identity in $Mod(L^*, \equiv_l, \equiv_w)$ the natural transformation

$$(1_{i(M(D))}, 1_{o(M(D))}),$$

where both epi multi-morphisms are defined using the identity relation in Ω .

The usual limits and colimits in $Mod(L^*, \equiv_l, \equiv_w)$ are computed based on that are made in the category of Ω -sets and epi multi-morphism, $epi-Set(\Omega)$. Where the product (in usual sense) exists for two Ω -sets $\alpha : A$ and $\beta : B$, if

$$\bigvee_{a, a'} \alpha(a, a') = \top \text{ and } \bigvee_{b, b'} \beta(b, b') = \top,$$

and it is defined by object $\alpha \otimes \beta : A \times B$ and the usual projections $\pi_A : A \times A \rightarrow A$ and $\pi_B : B \times B \rightarrow B$ in Set and codified as a function in $Set(\Omega)$. Note that, for instance, π_A is a epi multi-morphism, since $\pi_A^\circ \otimes (\alpha \otimes \beta) \otimes \pi_A$ define the multi-diagram, presented in fig. 11,

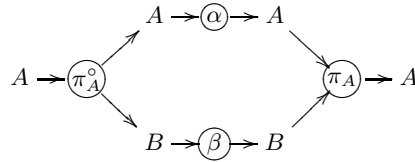


Fig. 11. Multi-morphism $\pi_A^\circ \otimes (\alpha \otimes \beta) \otimes \pi_A$.

by composition we have

$$\bigvee_{b'} \bigvee_{b''} \bigvee_{a'} \bigvee_{a''} \pi_A^\circ(a, b', a') \otimes \alpha(a', a'') \otimes \beta(b', b'') \otimes \pi_A(a'', b'', a''') =$$

$$\begin{aligned}
&= \bigvee_{b'} \bigvee_{a'} \pi_A^\circ(a, b', a') \otimes \bigvee_{a''} \bigvee_{b''} (\alpha(a', a'') \otimes \beta(b', b'') \otimes \pi_A(a'', b'', a''')) = \\
&= \bigvee_{b'} \pi_A^\circ(a, b', a) \otimes \bigvee_{b''} (\alpha(a, a''') \otimes \beta(b', b'') \otimes \pi_A(a''', b'', a''')) = \\
&= \bigvee_{b'} \bigvee_{b''} (\alpha(a, a''') \otimes \beta(b', b'')) = \alpha(a, a''') \otimes \bigvee_{b'} \bigvee_{b''} \beta(b', b'') = \alpha(a, a''')
\end{aligned}$$

The category $epi\text{-}Set(\Omega)$ doesn't has an initial object. However, for Ω -object $\alpha : A$ with a unique factorization $\alpha = f^\circ \otimes f$, the multi-morphism $f : \emptyset \rightarrow A$ is the only epi multi-morphism, since $f^\circ \otimes f = \alpha$. But there is only one epi multi-morphism to $\emptyset : \emptyset$ given by the empty relation $\emptyset : A \rightarrow \emptyset$ since $\emptyset^\circ \otimes \alpha \otimes \emptyset = \emptyset$.

Given a pair of epi multi-morphisms $f, g : A \rightarrow B$ from $\alpha : A$ to $\beta : B$. There is a equalizer for f and g , and it is given by $\gamma : A$ if and only if

$$(f \otimes g)^\circ \otimes \gamma \otimes (f \otimes g) = \beta.$$

Every pair of epi multi-morphisms f and g has a coequalizer and it is given by $\gamma : B$ such that

$$\gamma := (f \otimes g)^\circ \otimes \alpha \otimes (f \otimes g).$$

And, every family $(f_i : \alpha_i \rightarrow \beta)$ of epi multi-morphisms have wild pushouts given by the product $\otimes_i \alpha_i : \prod_i A_i$ and its projections $\pi_j : \prod_i A_i \rightarrow A_j$. Since, $\pi_j^\circ \otimes_i \alpha_i \pi_j = \alpha_j$ and $f_j^\circ \otimes \pi_j^\circ \otimes_i \alpha_i \pi_j \otimes f_j = f_j^\circ \otimes \alpha_j \otimes f_j = \beta$. Because $epi\text{-}Set(\Omega)$ have coequalizers and wild pushouts it has connected colimits (see [3]). By definition of natural transformation in $Mod(L^*, \equiv_l, \equiv_w)$ we have:

Proposition 12 *The category $Mod(L^*, \equiv_l, \equiv_w)$ has connected colimits in the usual sense.*

Since $epi\text{-}Set(\Omega)$ and $Mod(L^*, \equiv_l, \equiv_w)$ have connected colimits they have directed colimits, i.e. exist colimit for diagrams like $D : (I, \leq) \rightarrow epi\text{-}Set(\Omega)$ where (I, \leq) is a poset and if the vertices are models then its colimit is a model (see definition in [11]).

Following [11], a category is accessible, provided that has directed colimits and has a set \mathcal{A} of presentable objects such that every object is a direct colimit of objects from \mathcal{A} . And accessible categories can be characterized by:

Proposition 13 [11] *Each small category with split idempotents is accessible*

Where, a category has split idempotents if for every morphism $f : A \rightarrow A$ with $f.f = f$ there exist a factorization $f = i.p$ where $p.i = id_A$.

Example 13 *If Ω is a ML-algebra with at least three logic values $\perp < \lambda < \top$. The multi-morphism*

$$f = \begin{bmatrix} \top & \alpha \\ \alpha & \top \end{bmatrix}$$

is an epi multi-morphism and is idempotent but non-splittable, since for every factorization $f = i \otimes p$, $p \otimes i \neq id$ by proposition 3.

The fact described by this example are the rule in $epi\text{-}Set(\Omega)$ and $Mod(L^*, \equiv_l, \equiv_w)$. Since most of the idempotents aren't split idempotents they have:

Proposition 14 *Let Ω is a ML-algebra with more than two logic values. Then categories of models of libraries $Mod(L^*, \equiv_l, \equiv_w)$, aren't accessible.*

This means that, we can't use Ehresman sketches to specify the category of models of a library [11], i.e. model categories can't be axiomatizable by basic theories in first-order logic.

8 Sign systems and Semiotics

Lets now find what is the basic structure need on a library to define useful fuzzy structures.

Example 14 (Signatures as libraries) *Let Σ be a signature. By example 8 a signature can be seen as a library. The set $T(\Sigma)$ of terms defined by Σ is given, in [16], as the least set satisfying:*

1. each variable x is in $T(\Sigma)$, and if it has type b we write $x : b$;
2. if $f \in \Sigma$ with $ar(f) = (< a_i >_I, b)$ and $< t_i >_I$ is a list of terms from $T(\Sigma)$, with $t_i : a_i$ for every $i \in I$, then $f(t_i)_I$ is in $T(\Sigma)$, and it has type b , and we write in this case $f(t_i)_I : b$.

A term is closed if it doesn't contain variables. We have

$$T(\Sigma) \cong \text{Lang}(L(\Sigma, S))$$

if and only if we impose the existence of:

1. especial symbols c and v in S , as described in example 8 ;
2. diagonal components \triangleleft in $|L(\Sigma, S)|$ with $i(\triangleleft) = a$ and $o(\triangleleft) = < a >_I$, in $L(\Sigma)$, one for each alphabet symbol a in $|L(\Sigma, S)|$ and each "class" of chain equivalences I . These components represent dependencies in the term structure inherent to the use of the same variable more than once on the term definition.

If t is a term in $T(\Sigma)$ then it can be represented as a multi-graph homomorphism

$$t : \mathcal{G} \rightarrow \mathcal{G}(L(\Sigma, S)).$$

This graph homomorphism is usually called the parsing graphs for t . If variables involved on the definition of a term are different then the diagram t shape, \mathcal{G} , is a tree. On fig. 12 we present allowed configurations defining terms

$$f(x : b, g(y : c, z : d) : e) : a \text{ and } f(x : b, g(y : c, x : b) : e) : a.$$

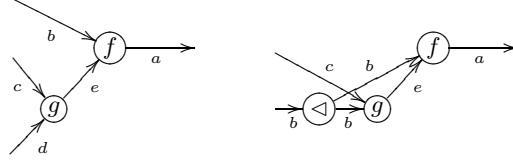


Fig. 12. Multi-morphism $f(x : b, g(y : c, z : d) : e) : a$ and $f(x : b, g(y : c, x : b) : e) : a$.

Let now Σ be a signature with a special collection of functional symbols denoted by $=_a$, one for each data type symbol a used on the signature, where

$$ar(=_a) = (< a, a >, l).$$

In the arity of $=_a$, the symbol l must be seen as an identification for the set of truth-values and should have associated two constant operators T and F , both with arity $(< >, l)$, used to identifying the true and the false on the associated logic framework. By practical reasons, given two terms of type a , $t : a$ and $s : a$, instead of writing the term $=_a(t, s) = T$, we use the usual infix notation and write $t =_a s$ calling it an equation of type a . In a signature with this characteristics we called relational symbol to every functional symbol f with arity of type

$$ar(f) = (< a_i >_I, l).$$

In this sense the symbol $=_a$ is relational, and this sort of signature Σ is said to have a logic structure.

A formula f , on a signature Σ with a logic structure, is a term which has by representation a multi-graph homomorphism having by output a relation, i.e. $o(f) = < l >$.

The above example requires the existence in Σ of a special symbol " l " and the existence of special component labels " $=$ " and " $<$ ", having predefined interpretations. This necessity can be seen frequently in other examples. For this type of labels we will define restrictions to the language model structures by fixing interpretation to some signs and to some structures definable in the library. We specify this type of structures using Ehresmann sketches defined by multi-graphs. We called specification systems to this generalization.

Definition 14 A specification system S , using a library L (or a sign system S using L) is a structure $S = (L, \mathcal{E}, \mathcal{U}, co\mathcal{U})$ where

1. $\mathcal{E} \subset Lang(L)$ is a set of **finite diagrams**, interpreted as a total multi-morphism (see definition 4),
2. \mathcal{U} is a set of tuples $(f, D, i(D), o(D))$ where f is a component and D is a **finite configurations**, such that f is interpreted as the multi-morphism defined by the limit of D having sign $i(D)$ as input vertices and sign $o(D)$ as output vertices (see definition 12), and

3. $co\mathcal{U}$ is a set of tuples $(f, D, i(D), o(D))$ where f is a component and D is a **finite configurations**, such that f is interpreted as the multi-morphism defined by the colimit of D from $sign\ i(D)$ to $o(D)$ (see definition 13).

In a specification system $S = (L, \mathcal{E}, \mathcal{U}, co\mathcal{U})$, while the set \mathcal{E} define structural proprieties to be preserved by its models, sets \mathcal{U} and $co\mathcal{U}$ impose restrictions to the structure for sign interpretations.

Extending the definition of model of a small Ehresmann sketches to interpretations of sign systems in $Set(\Omega)$ we have:

Definition 15 (Model for a specification system) *A model M in $Mod(L^*, \equiv_l, \equiv_w)$, for library L , is a model for the specification system $S = (L, \mathcal{E}, \mathcal{U}, co\mathcal{U})$ if:*

1. *for every pair $D \in \mathcal{E}$, $M(D)$ is a total multi-morphism;*
2. *for every $(f, D, i(D), o(D)) \in \mathcal{U}$, $M(f)$ is the multi-morphism defined by $Lim\ MD$ from $M(i(D))$ to $M(o(D))$;*
3. *for every $(f, D, i(D), o(D)) \in co\mathcal{U}$, $M(s)$ is the multi-morphism defined by $coLim\ MD$ from $M(i(D))$ to $M(o(D))$.*

The category defined by models for a specification system in $Set(\Omega)$ and natural transformations between interpretations is denoted by $Mod(S)$. And we call *the sketch structure of S* to $(\mathcal{E}, \mathcal{U}, co\mathcal{U})$. By definition, the category $Mod(S)$ is a full subcategory of $Mod(L^*, \equiv_l, \equiv_w)$. And, since the category of library models don't have split idempotents relations evaluated in Ω :

Proposition 15 *Let Ω be a not trivial ML-algebra with more than two truth values. Given a specification system S , the category $Mod(S)$ isn't an accessible category.*

Since the category Set is a full subcategory of $Set(\Omega)$ each map can be seen as a relation and because we can codify commutative diagrams using total morphisms and limit cones using the limit structure, defined in 9, trivially we have:

Proposition 16 *Every model for a Ehresmann Limit sketch in Set , defined using finite diagrams, can be specified by a sign systems in $Set(\Omega)$ for every not trivial ML-algebra Ω .*

By this, every algebraic theory (see definition in [11]) has a fuzzy version defined in $Set(\Omega)$.

Since in the following we will work over models of specification systems we define a semiotic as a sign system furnished with a model. This structure associates syntactic and semantic components to a language on the Goguen's institution spirit [13]. Formally

Definition 16 (Semiotic system) *A semiotic system is a pair (S, M) defined by a sign system $S = (L, \mathcal{E}, \mathcal{U}, co\mathcal{U})$ and a model $M \in Mod(S)$.*

We will denote by $Lang(S)$ the language associated to a sign system $S = (L, \mathcal{E}, \mathcal{U}, co\mathcal{U})$ or associated to a semiotic system (S, M) .

In the context of information systems: we can see a system specification as a database structure and a semiotic defined with this structure as a database state. Each database update induces a change in the database state, implying a semiotic change since it reflects a change in the system attributes relations codified on the database tables. The information system is then a semiotic since it is usually defined as a database instance or state. Then the information stored in the information system can be queried in the associated semiotic.

Lets see an example of a semiotic and how we can query it using limits.

Example 15 [21] *For the IDA'01 - Robot Data Challenge - series of vectors of binary data was generated by the perceptual system of a mobile robot. We suspect the generated time series contains several patterns (where a pattern must be seen as a structure in the data that is observed, completely or partially, more than once) but we not know the pattern boundaries, the number of patterns, or the structure of patterns. We suspect that at least some patterns are similar, but perhaps no two are identical. The challenge is to find the patterns and elucidate their structure. A supervised approach to the problem might involve learning to recognize patterns given known examples of patterns.*

The robot dataset is a time series of 22535 binary vectors of length 9, generated by a mobile robot as it executed 48 replications of a simple approach-and-push plan. In each trial, the robot visually located an object, oriented to it, approached it rapidly for a while, slowed down to make contact, and attempted to push the object. In one block of trials, the robot was unable to push the object, so it stalled and backed up. In another block the robot pushed until the object bumped into the wall, at which point the robot stalled and backed up. In a third block of trials the robot pushed the object unimpeded for a while. Two trials in 48 were anomalous.

Data from the robot's sensors were sampled at 10Hz and passed through a simple perceptual system that returned values for nine binary variables. These variables indicate the state of the robot and primitive perceptions of objects in its environment. They are: STOP, ROTATE-RIGHT, ROTATE-LEFT, MOVE-FORWARD, NEAR-OBJECT, PUSH, TOUCH, MOVE-BACKWARD, STALL. For example, the binary vector [0 1 0 1 1 0 1 0 0] describes a state in which the robot is rotating right while moving forward, near an object, touching it but not pushing it. Most of the 512 possible states are not semantically valid, however the robot's sensors are noisy and its perceptual system makes mistakes.

The dataset was segmented into episodes by hand. Each of 48 episodes contains zero or more instances of seven episode types, labelled A, B1, B2, C1, C2, D and E. The entire corpus contains 356 instances of these episode types.

We may use domain knowledge to define a library L which can be used to characterize relations between attributes. The easier way to do this library is by directly specifying its parsing graphic $\mathcal{G}(L)$. For that we fixed as signs Move, Objects, Path, Node, Episode, Rotate, Stalled and Class and by specifying compo-

nents pushing, direction, $stat_1$, proximity, source, target, start, end, direction, $stat_2$ and type having its constraints defined in the graph below.

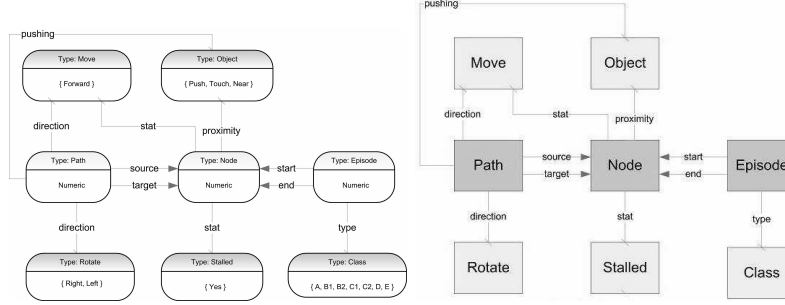


Fig. 13. Parsing Graph and example of a query.

This structure defines a semiotic when we assign to it a model by fixing an interpretation to each sign and for each component. Note that the defined semiotic is consistent with the available data if there is a word in the associated graphic language having as part of its the given dataset. However, for practical proposes, the lack of expressive power for the used language make this notion of consistence to restrictive. We relaxed this by defining what we mean by a semiotic λ -consistent with the data: a specification expressed in the graphic language, is λ -consistent with the data, if there is an interpretation having a part that is "good approximation" to the given data.

A semiotic selected for the parsing graph from fig. 13 have the signs interpretation domains equipped with a similarity relation. The limit for the diagram in fig. 13, where we identify the diagram sources $\{\text{Move, Object, Rotate, Stalled}\}$, the target $\{\text{class}\}$ and as auxiliary signs $\{\text{Path, Node, Episode}\}$, is a "good" approximation to the dataset. This limit can be seen as a Ω -set

$$\alpha : \text{Move} \times \text{Object} \times \text{Rotate} \times \text{Stalled} \times \text{Class}.$$

The discrepancies between α and the real data must then be seen as information that are not semantically valid for defined semiotic. This type of limit can be seen as a view of the data described by the semiotic. However the information in the generated limit isn't adequate to be used for solve the proposed problem of patterns detection, using machine learning algorithms. It doesn't codify the structure of time series generated by the robot perceptual system, since it doesn't use temporal relation between stats.

We used limits of admissible configuration to extract potential useful information from the universe modeled by the semiotic. The existence of patterns associate to robot stall on first three states of each episode should be detected in

the limit for diagram $D(a)$ in fig. 14, using the adequate machine learning tools. However, to classify episodes it seems to be more relevant the last robot stats.

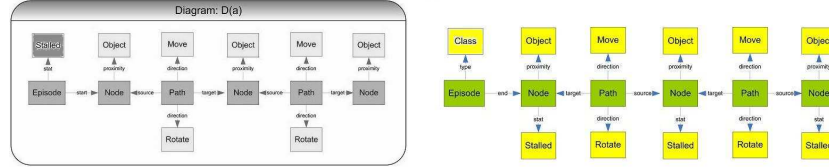


Fig. 14. Queries $D(a)$ and $D(b)$.

Patterns of information associated to the last three states of a robot are present in the limit for diagram $D(b)$ from fig. 14.

If we wish to use, for episode class prediction, the relational information available for three consecutive states we must change used library. We have a problem, the defined library structure doesn't have sufficient expressive power to describe this type of query. We may improve the library expressive power by adding to the specification system more a component: associating each automata stat to its episode. The query can be defined by diagram $D(c)$ in fig. 14.

The update of libraries can be made also to restricted its interpretations. In fig. 15 we enriched the sign system by imposing a restriction to its interpretation by adding two equalizers. We used them to impose that in a path the source and the target must be different. This can be codify by restricting the equalizer between components source and targets to a initial relation. If we want also consider only episode having more than a stat the limit of diagram define by component start and end must be the initial relation. The specification bellow is enriched also with new components $Lim D(a)$, $Lim D(b)$, $Lim D(c)$ and $Lim D(a)$ interpreted as the limit of presented queries $D(a)$, $D(b)$, $D(c)$ and $D(d)$, respectively, add as signs interpreted as the source of this new components. This new signs add a new level to the sign ontology. They are more general than the signs defined initially.

Note what the limit of diagram $D(c)$, presented in fig. 14, describe available information about signs having by interpretation three consecutive stats, and the limit of $D(d)$ describes three consecutive stats of episodes such that the robot stall.

In the example we used limits as a way to query the structure of a semi-otic. On the following we will formalize some of the concepts presented in this example, particularly that we mean by an "approximation" to the given data.

Example 16 (Neural Networks [27]) A neural network is a network of simple processing elements (neurons), which can exhibit complex global behavior,

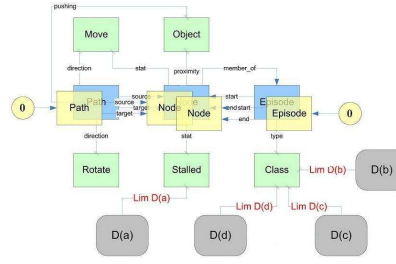


Fig. 15. Enriched sign system.

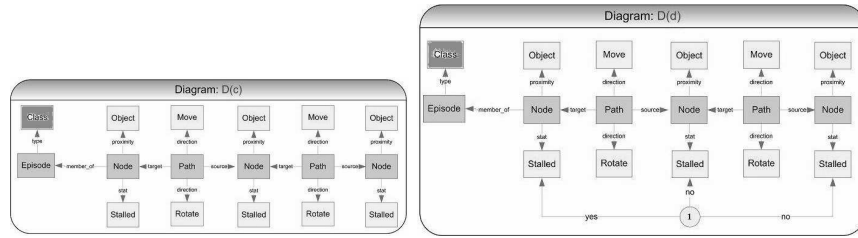


Fig. 16. $D(c)$ querying three consecutive stats and $D(d)$ querying three consecutive stats of episodes where the robot stall.

determined by the connections between the processing elements and element parameters. Formally, a network is a function f defined as a composition of other functions $g_i(x)$, with can further be defined as a composition of other functions. This dependencies can be conveniently represented as a network structure, with arrows depicting the dependencies between variables. What has attracted the most interest in neural network is the possibility of be parameterized to learning a task. Given a specific task to solve, and a class of functions F defined using a network structure and a class of neurons, learning means using a set of observations, in order to find $f^* \in F$ with solves the task in an optimal sense.

In this sense we can see an artificial Neural Network as an admissible diagram defined in the semiotic codifying every possible parametrization of each processing element. More precisely, the associated sign system describe the possible Neural Network structures and a model for it represents a set of parameterizations describing the behave of each processing element.

There are three major learning paradigms, each corresponding to a particular learning task [2]. These are supervised learning, unsupervised learning and reinforcement learning and can be seen as different ways of search the space defined by models of admissible configurations in order to find the optimal solution, i.e. the model what best fits the data.

Given a diagram D on a Neural Network Semiotic the multi-morphism $\text{Lim } D$ describe the functional behavior for the network D when applied to its input

space. In this sense, a network D learned a concept describe in a dataset if part of its limit $\text{Lim } D$ is a good approximation to the dataset.

More examples can be taken from applications of generic programming, see for instance [12].

9 Logics

We can see a semiotic as a formal way to specify a problem Universe of Discourse. We are particularly interest on the specification of universes where its entities can be characteristics by propositions on monoidal logics. For that we define:

Definition 17 *A semiotic system (S, M) , with*

$$S = (L : |L| \rightarrow (\text{Chains} \downarrow \Sigma^+), \mathcal{E}, \mathcal{U}, \text{col}\mathcal{U}),$$

is a logic semiotic system, if:

1. *exists a sign in S interpreted as the support Ω of a ML-algebra having as operators interpretations of component labeled with the signs $\vee, \wedge, \otimes, \Rightarrow, \perp$ and \top ,*
2. *for every string $w = s_0 s_1 \dots s_n$ of sign in S there is a label $=_w$ interpreted by M as the similarity $\bigotimes_i^n [\cdot = \cdot]_i$ where $[\cdot = \cdot]_i$ is the similarity on the Ω -set $M(s_i)$, defined in 8,*
3. *for every sign s in S and every natural number n there is a component in S labeled by \triangleleft_s^n and interpreted by M as a diagonal*

$$\triangleleft_{M(s)}^n : M(s) \rightarrow \prod_{i=1}^n M(s), \text{ and}$$

given by

$$\triangleleft_{M(s)}^n(a, a_1, a_2, \dots, a_n) = \bigotimes_i^n [a = a_i].$$

4. *for every sign s in S and every natural number, n there is a component in S labeled by \triangleright_s^n and interpreted by M as a codiagonal relation*

$$\triangleright_{M(s)}^n : \prod_{i=1}^n M(s) \rightarrow M(s),$$

given by

$$\triangleright_{M(s)}^n(a_1, a_2, \dots, a_n, a) = \bigotimes_i^n [a_i = a].$$

5. we suppose the existence of a disjoint decomposition for the set of signs Σ , given by Σ_{aux} and Σ_{pri} , where signs in Σ_{aux} are called auxiliary, and for every pair $(s, u) \in \Sigma_{pri} \times \Sigma_{aux}$ there are components $r(s, u) : su^+$ and $r(u, s) : us^+$ in L , called rename component, and interpreted by M as the identity in $M(s)$, i.e.

$$M(r(s, u)) = id_{M(s)} \text{ and } M(r(u, s)) = id_{M(s)}.$$

In a semiotic logic the signs \triangleleft and \triangleright are used to relate together similar component inputs and similar components outputs. Rename components are used as a mechanism to codify the wires or links between components inputs and outputs on the diagram.

As usual, equations can be specified by commutative diagrams, in fig. 17 we specify the property $e_s + x = x + e_s = x$ (existence of identity e_s) using a commutative diagram. A model M for an additive library $L_A(S, C)$ defines an additive operator with identity if the diagram limit defines a total multimorphism.

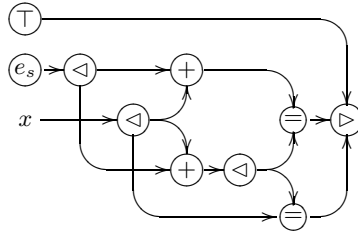


Fig. 17. Diagrams codifying $e_s + x = x + e_s = x$.

This diagram presented in fig. 17 can be codified in string base notation by the chain of signs:

$$\triangleleft_s^3 r(s, x) r(s, x) r(s, x) e_s \triangleleft_s^2 r(s, y) r(x, s) + r(s, z) r(x, s) r(y, s) + \triangleleft_s^2 r(s, w) r(s, w) r(z, s) r(w, s) = r(w, s) r(x, s) = \top \triangleright_\Omega^3.$$

By this we want to say that every diagram defining a word in a graphic language can be codified using string-based notation using rename components.

Example 17 Let $Set(\Omega)$ be defined using the ML-algebra $\Omega = ([0, 1], \otimes, \Rightarrow, \vee, \wedge, 0, 1)$, where $([0, 1], \otimes, \Rightarrow)$ is a model for product logic and $([0, 1], \vee, \wedge, 0, 1)$ the usual complete lattice defined in $[0, 1]$ by relation "less than". The Ω -set defined in $A = \{0, 1, 2\}$ by the similarity relation given by the table

$[- = -]$	0	1	2
0	1.0	0.5	0.0
1	0.5	1	0.5
2	0.0	0.5	1.0

and the additive relational operator

$- + 0$	$0 \quad 1 \quad 2$	$- + 1$	$0 \quad 1 \quad 2$	$- + 2$	$0 \quad 1 \quad 2$
0	$1.0 \ 0.5 \ 0.0$	0	$0.5 \ 1.0 \ 0.5$	0	$0.0 \ 0.5 \ 1.0$
1	$0.5 \ 1 \ 0.5$	1	$0.0 \ 0.5 \ 1.0$	1	$1.0 \ 0.0 \ 0.5$
2	$0.0 \ 0.5 \ 1.0$	2	$1.0 \ 0.5 \ 0.5$	2	$0.5 \ 1.0 \ 0.0$

define a model for an additive library and the operator have by identity $e_s = 0$, since the diagram in fig. 17 have by limit

A	Ω
0	$1.0=[0]$
1	$1.0=[1]$
2	$1.0=[2]$

In a logic semiotic system (S, M) the language $Lang(L)$, is called a *logic language*. Logic semiotics have sufficient expressive power to distinguish between diagrams defining relations and diagrams defining entities. For that, we classify the words as:

Definition 18 (*Graphic relations*) A diagram $D \in Lang(S)$, for a logic semiotic (S, M) , is called a relation when its output $o(D)$ is interpreted by M as the set of truth values Ω on $Set(\Omega)$.

Definition 19 A relation D is called an equation if diagram D can be decomposed as

$$D = I \otimes D_0 \otimes D_1 \otimes '=' ,$$

where $I = \triangleleft_{s_1}^{n_1} \dots \triangleleft_{s_m}^{n_m}$ is defined through realizations of diagonals, D_0 , and D_1 , are diagrams satisfying $o(D_0) = o(D_1)$ and $'='$ is a diagram defined using the unique component, interpreted as a similarity relation, and satisfying $i(' = ') = o(D_0).o(D_1)$. In this case we simplify notation by denoting the diagram D as $D_0 = D_1$. Note what, diagram I codifies the dependencies between interpretations of signs used as inputs for diagram $D_0 \otimes D_1 \otimes '='$, relating together signs having similar values.

In definition diagram $I = \triangleleft_{s_1}^{n_1} \otimes \dots \otimes \triangleleft_{s_m}^{n_m}$ captures in a graphic logic the dependence relations defined on string-based logic by repeating bounded variables on a proposition. The relation $D \in Lang(S)$ is called *true* by M if the limit $M(D \otimes \top \otimes \triangleright_{\Omega}^2)$ is a total multi-morphism. In this sense a equation D is universal if the interpretation of $D.\top.\triangleright_{\Omega}^2$ by M is a total multi-morphism.

Given a logic semiotic system (S, M) , let $Lang_R(S, M)$ be the subcategory of $Lang(S)$ having by objects diagrams defining relations. Using the operation of diagram gluing we define for pairs of relations $D_0, D_1 \in Lang_R(S, M)$ the following operators between diagrams:

1. $D_0 \otimes D_1$ is the diagram $I \otimes D_0 \otimes D_1 \otimes '\otimes'$,
2. $D_0 \Rightarrow D_1$ is the diagram $I \otimes D_0 \otimes D_1 \otimes '\Rightarrow'$,

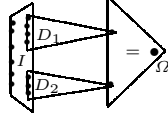


Fig. 18. Sketch for a multi-morphism of type $D = I \otimes D_0 \otimes D_1 \otimes ' \subseteq '$.

3. $D_0 \wedge D_1$ is the diagram $I \otimes D_0 \otimes D_1 \otimes ' \wedge '$ and
4. $D_0 \vee D_1$ is the diagram $I \otimes D_0 \otimes D_1 \otimes ' \vee '$,

where I is defined through realization of diagonals, linking together inputs having the some meaning by M . This allows the definition of new relations from pairs of simplest ones, and they correspond to the lifting part of Ω structure to diagrams in $Lang_R(S, M)$.

In the following we present some useful examples of logic semiotics important to characterize the expressive power of language used by some machine learning algorithms:

Example 18 (Binary semiotic $S_B(S, C)$) A binary semiotic is a logic semiotic where sets S and C define a binary library $L_B(S, C)$ (presented in example 9). We call to this sort of semiotic dataset semiotic or table semiotic since we can use instantiations of relations in $Lang_R(S_B(S, C))$ to codify datasets or tables.

The use of binary semiotic can be seen in machine learning algorithm used to generate decision rules like Apriori described in [27].

Example 19 (Linear semiotic $S_L(S, C)$) A linear semiotic $S_L(S, C)$ extends a binary semiotic. It is defined by a linear library $L_L(S, C)$ (presented in example 10) where \geq : ssl^+ is interpreted as a partial order in the interpretation of signs s , $M(s)$, for all symbol $s \in S$. This type of relation is codified in the model of a linear semiotic $S_L(S, C) = (L, \mathcal{E}, \mathcal{U}, col\mathcal{U})$ if it includes in the set \mathcal{E} the diagrams presented in fig. 19, codifying propositions represented in string-based first-order logic by

$$\forall x : x \geq x, \quad \forall x, y : (x \geq y \wedge y \geq x) \Rightarrow (x = y)$$

and

$$\forall x, y, z : (x \geq y \wedge y \geq z) \Rightarrow (x \geq z)$$

and are "preserved" by its models.

Lets present an example on $Set(\Omega)$ having Ω the a structure of ML-algebra where $([0, 1], \otimes, \Rightarrow)$ is a model for product logic and $([0, 1], \vee, \wedge, 0, 1)$ the usual complete lattice defined in $[0, 1]$. For the Ω -set defined with support $A = \{0, 1, 2, 3, 4\}$ and the similarity relation

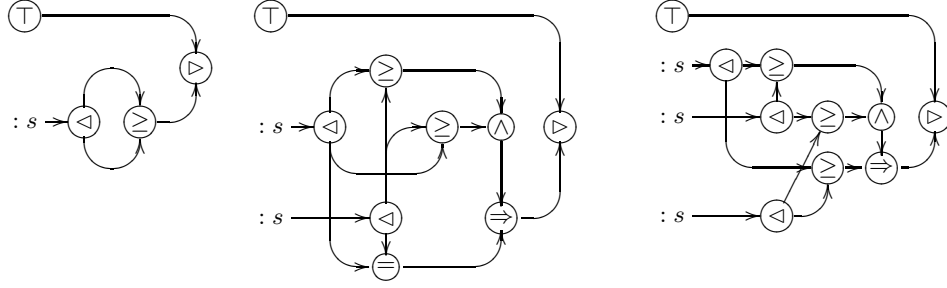


Fig. 19. Diagrams codifying $\forall x : x \geq x$, $\forall x, y : (x \geq y \wedge y \geq x) \Rightarrow (x = y)$ and $\forall x, y, z : (x \geq y \wedge y \geq z) \Rightarrow (x \geq z)$.

$[- = -]$	0	1	2	3	4
0	1.0	0.5	.25	0.0	0.0
1	0.5	1.0	0.5	.25	0.0
2	.25	0.5	1.0	0.5	.25
3	0.0	.25	0.5	1.0	0.5
4	0.0	0.0	.25	0.5	1.0

and the relational operator defined from A to A by:

$[- \geq -]$	0	1	2	3	4
0	1.0	0.5	.25	0.0	0.0
1	1.0	1.0	0.5	.25	0.0
2	1.0	1.0	1.0	0.5	.25
3	1.0	1.0	1.0	1.0	0.5
4	1.0	1.0	1.0	1.0	1.0

When this relations are used for the sign interpretation in the three diagrams presented on fig. 19 they have by limit, respectively,

$A \Omega$	$A \times A \Omega$		$A \times A \times A \Omega$
0 1	(0,0) 1	and	(0,0,0) 1
1 1	(1,0) 1		(1,0,0) 1
2 1	\vdots \vdots		\vdots \vdots
3 1	(3,4) 1		(3,4,4) 1
4 1	(4,4) 1		(4,4,4) 1

Which grants the commutativity of each diagram.

We call to this type of semiotics **grid semiotics**. They appear associated algorithms of machine learning used to generate decision rules like the C4.5Rules of J.R. Quinlan see [27].

Example 20 (Additive semiotic $S_A(S, C)$) A additive semiotic $S_A(S, C)$ is a linear semiotic $S_L(S, C)$ such that it is a additive library $L_A(S, C)$ (presented in example 11) where $(M(s), M(+ : sss^+))$ is a monoid, for all symbol $s \in S$,

and the interpretation for $e_s : s^+$ is the monoid identity, with $e_s \in C$. Monoid proprieties can be codified in the semiotic structure if the model transform each of the diagrams presented in fig. 20 in a total multi-morphism, for each sign s in the semiotic.

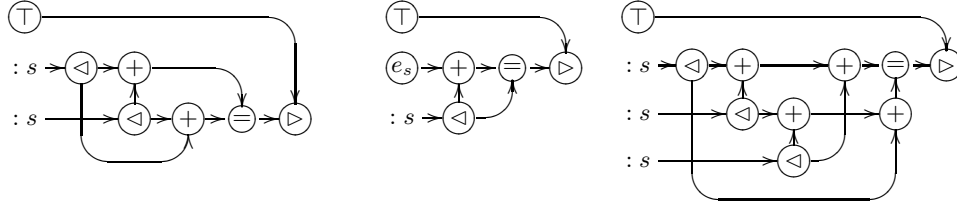


Fig. 20. Diagrams codifying $\forall x, y : x + y = y + x$, $\forall x : x + e_s = x$ and $\forall x, y, z : (x + y) + z = x + (y + z)$.

Example 21 (Multiplicative Semiotic $S_M(S, C)$) A multiplicative semiotic $S_M(S, C)$ is an additive semiotic $S_L(S, C)$ defined by a multiplicative library $\mathcal{A}_M(S, C)$ (presented in example 12) where $(M(s), \times)$ is a commutative semi-group, for all symbol $s \in S$.

In this semiotics we can codify rules defined using regression like the rules generated by machine learning algorithms like M5, of J.R. Quinlan, described in [22].

The definition of Domain of Discourse structure may impose restrictions to signs interpretations. In a binary semiotic we may impose rules for sign interpretation defined by Horn clauses of type:

$$(x_1 = c_1 \wedge x_2 = c_2 \wedge x_3 = c_3) \Rightarrow y = c_4.$$

The expressive power of linear semiotic allows the codification of rules like

$$(x_1 \leq c_1 \wedge x_2 \leq c_2 \wedge c_3 \leq x_1) \Rightarrow y \leq c_4,$$

and on multiplicative semiotic sign interpretation can be restricted to semiotics defined by models satisfying regression rules like:

$$(x_1 \leq c_1 \wedge x_2 \leq (c_3 \times x_1 + c_2) \wedge x_3 = c_3) \Rightarrow y \leq c_4 \times x_1 + c_5 \times x_2 + c_6 \times x_3 + c_7.$$

This type of regression rules can be codified by diagrams like the one represented on fig. 21, where frames present the obvious subdiagrams.

Models generated by some of the Data Mining and Machine Learning tools can be codified in one of this sign systems. By this we mean that we can extract

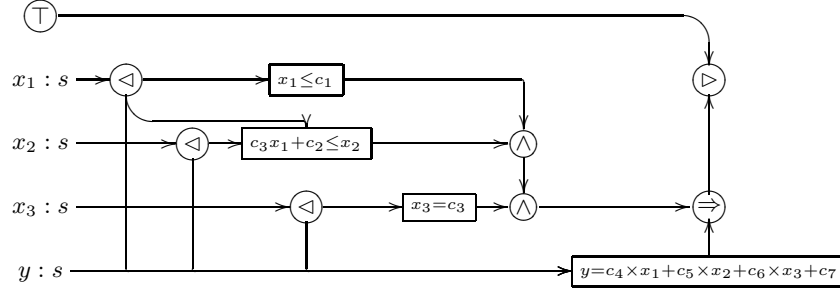


Fig. 21. Diagram codifying a regression rule.

rules from models generated by learning algorithms, what reflect the available data structure. This type of structure can be used on the enrichment of the sign system, employed to specify the information system, allowing the definition of constraints to its models compatible with, the existent data or views for, the reality. By this we want to say that the best description we may have for an information system is the best generalization available for the stored data.

10 Lagrangian syntactic operators

The expressive power of our specification languages can be increased using *Lagrangian syntactic operators* or *sign operators*. This operator are defined at the level of sign systems signs or/and components, and must be preserved via sign systems models, allowing the generation of new signs or components based on others signs or components.

An example of this operators, with evident applicability, are the differential operators. For that we define:

Definition 20 (Differential semiotic) *A logic semiotic system (S, M) is called a differential semiotic system if it is a multiplicative semiotic where exists a sign R interpreted as the support for a ring $(M(R), \times, +, 0, 1)$, and labels $\partial_w f$, in S , for components $f : i(f) \rightarrow R$ in S with output R , and w a word over its input symbols such that:*

1. $w \leq i(f)$,
2. $\partial_w f : d(f) \rightarrow R$,
3. and the following for component label defined below, using ' \times ' and ' $+$ ' on infix notation, we must have:
 - (a) if $f \equiv_l d_0 \times d_1$ then $M(\partial_w(d_0 \times d_1)) = M(\partial_w(d_0) \times d_1 + d_0 \times \partial_w(d_1))$,
 - (b) if $f \equiv_l d_0 + d_1$ then $M(\partial_w(d_0 + d_1)) = M(\partial_w(d_0) + \partial_w(d_1))$, and
 - (c) if $ww' \leq i(f)$ then $M(\partial_{ww'} f) = M(\partial_w(\partial_{w'} f)) = M(\partial_{w'}(\partial_w f))$

The component label operator ∂ allows the characterization of multi-morphisms impossible of represente on the associated logic semiotic. For instance, in a differential semiotic system we may interpret a component $f : xy \rightarrow R$ as a multi-morphism satisfying the conservative law when the following diagram is interpreted by the model as a total multi-morphism.

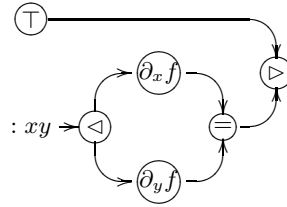


Fig. 22. Conservative law in a diferencial semiotic.

Lets see a naif application:

Example 22 (Modeling traffic) *In Lighthill-Whithams-Richards (LWR) model (presented in [18] and [25]), the traffic state is represented from a macroscopic point of view by the function $\rho(x, t)$ which represents the density of vehicles at position x and time t . The dynamics of the traffic are represented by a conservation law expressed as*

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho v}{\partial x} = 0,$$

where $v = v(x, t)$ is the velocity of cars at (x, t) . The main assumption of LWR model is that the drivers instantaneously adapt their speed in function of the surrounding density:

$$v(x, t) = V(\rho(x, t))$$

the function $f(\rho) = \rho V(\rho)$ is then the "flow rate" representing the number of vehicles per time unit. We have then

$$\frac{\partial \rho}{\partial t} + \frac{\partial f(\rho)}{\partial x} = 0.$$

The model is defined for a single unidirectional road. And it define a diferencial semiotic having the base library presented in fig. 23 and where the associated sign system have by total diagrams

$$\partial_t \rho + \partial_x f(\rho) = 0 \text{ and } 0 \leq \rho \leq \lambda,$$

where the least condition describes the road maximal density. A model for this sign system can be seen as an admissible car distribution on the road.

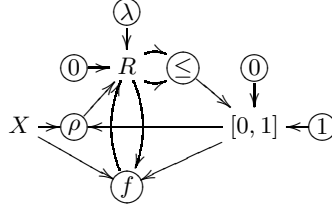


Fig. 23. The library for a sing unidirectional road model.

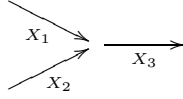


Fig. 24. A road network defined by the junction of two incoming roads and one outgoing road.

The above library doesn't has descriptive power to specify a road network. In fig. 25, we present an extension to the initial library. This new library allows the graphic modeling of a singles network defined by the junction of two incoming roads X_1, X_2 and one outgoing road X_3 with single direction. The semiotic of this problem is defined using the library from fig. 25 where we also add three

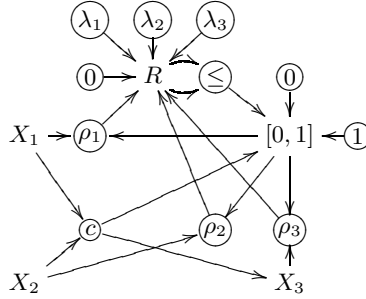


Fig. 25. The library for the presented road network.

"flow rate" components

$$f_1 : X_1 \times R \times [0, 1] \rightarrow R, \quad f_2 : X_2 \times R \times [0, 1] \rightarrow R \quad \text{and} \quad f_3 : X_3 \times R \times [0, 1] \rightarrow R$$

one for each roads and initial condition constants $\rho_{1,0}, \rho_{2,0}, \rho_{3,0} : \perp \rightarrow R$. Network traffic model restrictions are described by the following conservative laws and flow restrictions in each roads:

1. $\partial_t \rho_1 + \partial_x f_1(\rho_1) = 0$ and $0 \leq \rho_1 \leq \lambda_1$,
2. $\partial_t \rho_2 + \partial_x f_2(\rho_2) = 0$ and $0 \leq \rho_2 \leq \lambda_2$, and
3. $\partial_t \rho_3 + \partial_x f_3(\rho_3) = 0$ and $0 \leq \rho_3 \leq \lambda_3$.

In order to complete the model description, we define the mechanism that occurs at the junction. A first condition is the conservation of flows

$$f_1(\rho_1(0, t)) + f_2(\rho_2(0, t)) = f_3(\rho_3(0, t)), \forall t,$$

One of the elementary problem we can study, and from which a model exists, is the Riemann problem. For a Riemann problem at a junction, we take as initial condition a constant density on the three roads:

1. $\rho_1(x, 0) = \rho_{1,0}$,
2. $\rho_2(x, 0) = \rho_{2,0}$, and
3. $\rho_3(x, 0) = \rho_{3,0}$.

Models for this sign system may be totally unrealistic. For instance, constants, $\rho_1 = \lambda_1$, $\rho_2 = \lambda_2$ and $\rho_3 = 0$ can be associated to a possible semiotic, however this model although is clearly counterintuitive (except obviously in the presence of a red light at the entrance of the third road). A natural way to have a realistic model, for a particular road junction, is by adding rules describing the behavior of the drivers at the junction. We may enriched the sign system by adding extra information using models generated by machine learning algorithms for the data. This requires the integration of the defined semiotic with a semiotic associated to the language used in the description of machine learning model, problem described in the following.

11 Concept description

A *concept description*, using attributes (A_i) , on the logic semiotic system (S, M) is a Ω -map

$$d : \prod_i A_i \rightarrow \Omega,$$

where the family (A_i) is a sequence of Ω -sets. A concept description is defined in a semiotic (S, M) if the sequence (A_i) is defined using interpretation of signs in the language $Lang(S)$. For short, we write $d \in M(S)$ when we want to select a concept description in the semiotic (S, M) . Note what, d defines a relation in a monoidal logic and it may not be the interpretable by M of a relation in $Lang(S)$. Intuitively, a concept description can be seen as the state of knowledge about a concept at a given moment. A database specified by the sign system S can codify the concept d if there is a model $M \in Mod(S)$ and a diagram $D \in Lang(S)$ such that

$$M(D) = d.$$

In this case we say that the query D on the information system defined by semiotic (S, M) have by answer d .

Given two concept descriptions

$$d_0 : \prod_i A_i \rightarrow \Omega \text{ and } d_1 : \prod_i A_i \rightarrow \Omega,$$

we write $d_0 \leq d_1$ if $d_0(\bar{x}) \leq d_1(\bar{x})$ for every \bar{x} in $\prod A_i$. And we should note that every Ω -set A defines a concept description by the extend map $[\cdot] : A \rightarrow \Omega$. In this sense we will see every Ω -set as a concept description. And in $Set(\Omega)$ every Ω -set with support $\prod A_i$ have associated a complete lattice, of concept descriptions, having by bottom $\perp : \prod A_i \rightarrow \Omega$ and by top $\top : \prod A_i \rightarrow \Omega$. Particularly, the limit $M(D)$ is a concept description for every $D \in Lang(S)$.

Some concept descriptions can be codified by a semantic, others doesn't. Given a pair of concept descriptions

$$d_0 : \prod A_i \rightarrow \Omega \text{ and } d_1 : \prod A_i \rightarrow \Omega,$$

we define

$$\Gamma(d_0, d_1) = d_0 \Leftrightarrow d_1.$$

The biimplication Γ is a \otimes -similarity relation in $\prod A_i$ since:

1. $\Gamma(d_0, d_0) = \top$ (reflexivity)
2. $\Gamma(d_0, d_1) = \Gamma(d_1, d_0)$ (symmetry)
3. $\Gamma(d_0, d_1) \otimes \Gamma(d_1, d_2) = \Gamma(d_0, d_2)$ (transitivity) (by proposition 2)

When $\Omega = \{\perp, \top\}$ is a two valued logic, Γ is clearly an equivalence relation on $\prod A_i$.

Definition 21 Given a semiotic (S, M) , and a concept d . A diagram D of a λ -codification or a λ -description for d if

$$\Gamma(d, M(D)) \geq \lambda.$$

In this case we also say that MD is an approximation to the concept d . In this sense, relation D is an hypothesis describing the concept presented by d , selected on language $Lang(S)$.

This definition may be extended to concept descriptions having different support sets. Given concept descriptions

$$d_I : \prod_{i \in I} A_i \rightarrow \Omega \text{ and } d_J : \prod_{j \in J} A_j \rightarrow \Omega,$$

such that exist a projection map $\pi : \prod_{i \in I} A_i \rightarrow \prod_{j \in J} A_j$, we write $d_J \preceq d_I$, we call d_J a λ -projection of d_I if

$$\Gamma(\pi \otimes d_I, d_J) \geq \lambda.$$

Given a concept description d and an hypothesis D in $Lang(S)$ the quality of D as a description for d is given by:

$$[d = D] = \bigwedge_{\bar{x}} \Gamma(d, D)(\bar{x}),$$

where

1. $\Gamma(d, D)(\bar{x}) = (\pi \otimes M(D) \Leftrightarrow d)$ if $d \preceq M(D)$:
2. $\Gamma(d, D)(\bar{x}) = (M(D) \Leftrightarrow \pi \otimes d)$ if $M(D) \preceq d$.

We see a model as the fuzzy answers to a query on a semiotic for what we define:

Definition 22 *A concept d is a λ -model for D in $\text{Lang}(S)$ if $d \preceq M(D)$ or $M(D) \preceq d$ and the diagram presented in fig. 26 is a pullback such that*

$$\Gamma_\lambda(d, D) = \Pi A_i,$$

where $[\lambda, \top]$ is a chain on lattice Ω . In this case we write

$$d \models_\lambda \forall D.$$

$$\begin{array}{ccc} \Gamma_\lambda(d, D) & \xrightarrow{\quad} & [\lambda, \top] \\ \downarrow \subset & \lrcorner & \downarrow \subset \\ \Pi A_i & \xrightarrow{\Gamma(d, D)} & \Omega \end{array}$$

Fig. 26. Diagram evaluation.

If in the above pullback diagram we have

$$\Gamma_\lambda(d, D) \subseteq \Pi A_i \text{ and } \Gamma_\lambda(d, D) \neq \emptyset$$

we write

$$d \models_\lambda \exists D,$$

or, when we want be more formal,

$$d \models_\lambda \forall_B D,$$

where $B = \Gamma_\lambda(d, D)$. This notation is also used as $d \models_\lambda \forall_C D$ when $C \subseteq \Gamma_\lambda(d, D)$.

When $d \models_\top \forall D$, we write $d \models \forall D$, and d can be seen as the answer to the query D on the information system given by (S, M) . Similarly, if $d \models_\top \exists D$, we write $d \models \exists D$, part of d is λ -consistente with interpretation for D in the semiotic (S, M) .

Note what, $\forall_B D$ may not be seen as a formula on the language associated to the used semiotic. Because the language may not have sufficient expressive power to define B . However if

$$B = \Gamma_\lambda(d, D) = \Gamma_\top(\chi_B, D'),$$

i.e. if domain B can be specified using diagram D' in the language we write

$$d \models_{\lambda} \forall_{D'} D.$$

Note what, in this case, for every description d we have

$$d \models_{\lambda} \forall_{D'} D \Leftrightarrow d \models_{\lambda} \forall_{D'} \Rightarrow D.$$

When $d_0 \models_{\lambda_0} \forall_{B_0} D_0$ and $d_1 \models_{\lambda_1} \forall_{B_1} D_1$ we have

$$d_0 \otimes d_1 \models_{\lambda_0 \otimes \lambda_1} \forall_{B_0 \cap B_1} D_0 \otimes D_1,$$

$$d_0 \vee d_1 \models_{\lambda_0 \vee \lambda_1} \forall_{B_0 \cap B_1} D_0 \vee D_1,$$

$$d_0 \Rightarrow d_1 \models_{\lambda_0 \Rightarrow \lambda_1} \forall_{B_0 \cap B_1} D_0 \Rightarrow D_1.$$

From the proposed definition every diagram has a λ -model since:

Proposition 17 *In a logic semiotic (S, M) if D is a relation defined on language $Lang(S)$ then*

$$MD \models \forall D.$$

And from the presented notion of similarity, defined by biimplication, we have also as λ -models for D concepts λ -similar to its interpretation MD :

Proposition 18 *If $\Gamma(d, MD) \geq \lambda$ then $d \models_{\lambda} \forall D$.*

Naturally, we used the similarity definition to formalize what we mean by concepts consistent with relations.

Definition 23 (Consistence) *Given a semiotic (S, M) . A relation D from $Lang(S)$ is consistent with $d \in M(S)$ if $d \models \forall D$, and it is λ -consistent with d when $d \models_{\lambda} \forall D$. The relation D is consistent with part of d if $d \models \exists D$ and it is λ -consistent with a part of d when $d \models_{\lambda} \exists D$.*

The set of hypotheses consistent with d is denoted by $Hy_{(S, M)}(d)$. For every $\lambda \in \Omega$, the set of hypotheses λ -consistent with d is denoted by $\lambda Hy_{(S, M)}(d)$. And, for a chain of truth values in Ω

$$\top \geq \lambda_0 \geq \lambda_1 \geq \dots \geq \lambda_n,$$

we have

$$Hy_{(S, M)}(d) \subseteq \lambda_0 Hy_{(S, M)}(d) \subseteq \lambda_1 Hy_{(S, M)}(d) \subseteq \dots \subseteq \lambda_n Hy_{(S, M)}(d).$$

Example 23 (Description consistent with a dataset) *Let (S, M) be a binary semiotic having by signs A, B, C, D and let*

$$d : M(A) \times M(B) \times M(C) \times M(D) \rightarrow \Omega,$$

be a finite crisp concept description, i.e. $d(\bar{x}) = \top$ or $d(\bar{x}) = \perp$ for every entity \bar{x} , and the number of entities \bar{x} such that $d(\bar{x}) = \top$ is finite. Then there is a word D in the language, associated to the semiotic, consistes with d called the dataset used to describe d .

Let be more specific, suppose that signs A, B, C, D have the some interpretation, let $M(A) = M(B) = M(C) = M(D) = [0, 1]$. And suppose that:

1. $d(1.0, 0.5, 0.2, 0.2) = \top$
2. $d(1.0, 1.0, 0.2, 0.2) = \top$, and
3. $d(1.0, 1.0, 0.0, 0.2) = \top$

are the only tuples true in relation d . This relation is consistent with the diagram

$$(A=1.0 \otimes B=0.5 \otimes C=0.2 \otimes D=0.2) \otimes (A=1.0 \otimes B=1.0 \otimes C=0.2 \otimes D=0.2) \otimes (A=1.0 \otimes B=1.0 \otimes C=0.0 \otimes D=0.2)$$

or d is the answer to the query defined by the diagram, usually represented using table notation by:

A	B	C	D
1.0	0.5	0.2	0.2
1.0	1.0	0.2	0.2
1.0	1.0	0.0	0.2

Fig. 27. Dataset.

12 Fuzzy computability

When the interpretation of a diagram is consistent with a multi-morphism we consider the multi-morphism computable in the semiotic. Formally:

Definition 24 (Computability) *Given a semiotic (S, M) . A multi-morphism $f : A \multimap B$ is computable in (S, M) if there is a diagram D in $Lang(S)$:*

1. having A as input, $A = i(D)$, $B = o(D)$ by output, and
2. codify F , i.e. $f \models \forall D$.

The multi-morphism f is λ -computable in (S, M) if $A = i(D)$, $B = o(D)$ and $f \models_{\lambda} \forall D$. These notions are very restrictive. We relaxed them by calling to a diagram D a specification to compute part of f if $d \models \exists D$. When the domain of the computable part of f can be described by a diagram D' we write

$$f \models \forall_{D'} D.$$

When $f \models \forall D$, with $A = i(D)$ and $B = o(D)$, we call diagram D a *program* or a *specification*, in language $Lang(S)$, and its image by M is an implementation for $f : A \multimap B$.

In this sense every, and only, interpretation of words from $Lang(S)$ are computable in the semiotic (S, M) . And, since words in $Lang(S)$ are generated from atomic componentes we have:

Proposition 19 *If f and g are computable multi-morphisms in the semiotic (S, M) the $f \otimes g$ is also computable in (S, M) .*

And, since $Lang(S)$ is defined by finite diagrams, every finite diagram D in $Set(\Omega)$, having by arrows computable multi-morphisms, has by limit a computable multi-morphism.

The set of interpretations of words from $Lang(S)$ and computable multi-morphisms define a category, denoted by $Hy_{(S,M)}$. In this category we write $f : d_1 \rightarrow d_2$ if f is a computable multi-morphism and d_1 and d_2 are consistent, descriptions in the semiotic, satisfying $d_1 \otimes f = d_2$. Note what, if D is consistent with d_1 and D_f is the specification for f then the diagram $D \otimes D_f$ is consistent with $d_1 \otimes f$.

Generically, if $(d \Leftrightarrow MD) \geq \lambda$ and $(f \Leftrightarrow MD_f) = \top$ then $(d \otimes f \Leftrightarrow MD \otimes f) \geq \lambda$, i.e. $(d \otimes f \Leftrightarrow MD \otimes MD_f) \geq \lambda$. Formally:

Proposition 20 *Let d be a description λ -consistent with D and f a computable multi-morphism specified by D_f . Then $d \otimes f$ is a description λ -consistent with diagram $D \otimes D_f$.*

In this sense a computable multi-morphism is known as a pre-processing tool in the data mining community. This allows the definition of $\lambda Hy_{(S,M)}$, the category of concept λ -consistentes and computable multi-morphisms in the semiotic (S, M) . Naturally, the limit and the colimit, in the usual sense, of finite diagrams in $\lambda Hy_{(S,M)}$ define computable relations. We call this type of finite diagrams *mining schemas*. And, given a mining schema D , in semiotic (S, M) , and a λ -consistent concept d , the limit $Lim D$ defines a computable multi-morphism and $d \otimes Lim D$ is a λ -consistent concept, interpreted as the output of schema D when applied to concept d .

As usual we extend the notion of computability defining:

Definition 25 (Turing computable) *A concept d is called Turing computable in the semiotic (S, M) if there is a diagram D , possible infinite but enumerable, such that*

$$Lim D = d.$$

Computability is usually associated with state-based systems. The interpretation, in a semiotic, of a stat must be time dependent. Given the presented static definition of sign interpretation we only catch the dynamic beaver using an ontological hierarchy. We see the possible interpretation of a sign as a class of structures used as possible instantiation for it during the system execution. We achieved this using a syntactic operator linking together signs in a same class representing different views for the same entity. The class of related signs using the syntactic operator must have the same generalization sign in the sign ontology. The existence of this type of syntactic operator, in a semiotic, defines what we called a syntactic operator in section 10.

Definition 26 (Temporal semiotics) *A temporal semiotic is a semiotic (S, M) , defined by a library $L : |L| \rightarrow (Chains \downarrow \Sigma^+)$, and having a syntactic operator*

$$t : \Sigma^+ \rightarrow \Sigma^+$$

such that:

1. preserves polarization of signs, $t(s^+) = t(s)^+$, for every $s \in \Sigma$;
2. preserves concatenation, $t(w_0.w_1) = t(w_0).t(w_1)$ for every pair of words w_0, w_1 ;
3. preserves components functionality, if $f : w \rightarrow w'$, it must exists a component

$$t(f) : t(w) \rightarrow t(w').$$

We imposed the existence of a component

$$t(r) : i(t(w)) \rightarrow o(t(w))$$

for every component $r : i(w) \rightarrow o(w)$, and an ontological hierarch for signs time invariant relating time dependent sings, i.e. if $s_1 = t(s_0)$ then it must exist a sign s such that $s_1 \leq s$, $s_0 \leq s$ and $s = t(s)$. In this sense, every sequence of time dependent signs

$$s_0, t(s_0), t(t(s_0)), t(t(t(s_0))), \dots$$

have by generalization the same sign s on the ontology. We call s a time invariant sign.

In a temporal semiotic (S, M) , if $r : i(w) \rightarrow o(s(w))$ is a component in the semiotic then its interpretation $M(r) : M(i(w)) \rightarrow M(o(s(w)))$ is called a coalgebra. A sign $s \in \Sigma$ is *time-invariant* in the semiotic if $M(s) = M(t(s))$.

A *temporal logic semiotic* is a semiotic which is a logic semiotic and a temporal semiotic.

Example 24 (Fuzzy Turing machine) A fuzzy Turing machine, with tape define using signs from F , can be defined as a word in language associated to a temporal logic semiotic (S, M) . And the interpretation for this word can be seen as an execution for it. The machine structure can be codified in a sign system S with library L having by signs a set of machine stats, Q , and having by components the Turing machine instructions with labels in a set I .

Each of the instructions in I has conditional form: it tells what to do, depending on whether the symbols distribution being scanned (the distribution of symbols in the scanned square). Namely, there are three classes of things that can be done:

1. Print: Change signs distribution in place of whatever is in the scanned square;
2. Move one square to the right;
3. Move one square to the left;

So depending on what instruction is being carried out and on what distribution is being scanned, the machine or its operator will perform one or another of these actions.

An instruction define a link between two stats and are codified as component labels with the following structure.

1. $q_0[f]q_1$ if in stat q_0 the scanned distribution is changed using component f interpretation and then change to stat q_1 ;
2. $q_0[d_0 : L]_{\lambda} q_1$ if in stat q_0 is reading a distribution d and $d \otimes M(d_0) \geq \lambda$ then move left and change to stat q_1 ;
3. $q_0[d_0 : R]_{\lambda} q_1$ if in stat q_0 is reading a distribution d and $d \otimes M(d_0) \geq \lambda$ then move right and change to stat q_1 .

An instruction is executed if its condition is verified.

In this sense a diagram in $\text{Lang}(L)$, defined using signs time invariant, is a Turing machine specification with stats in Q and tape signs from F . Every refinement of a Turing machine specification in $\text{Lang}(L)$, defined using only time variant sings, is called a flow chart and codifies a Turing machine execution. However to garante the correct interpretation of an instruction we have, for each stat $q_i \in Q$ in the sign system, signs

$$q_i^{(r)}, q_i^{(m)}, q_i^{(l)}, q_i^{(hr)}, q_i^{(tr)}, q_i^{(hl)}, q_i^{(tl)}$$

where $q_i^{(r)}$ is interpreted as the tape right half, $q_i^{(m)}$ the reading square, $q_i^{(l)}$ is interpreted as the tape left half. And, for each tape halves we select the right half head $q_i^{(hr)}$, the right tail head $q_i^{(tr)}$, the left half head $q_i^{(hl)}$ and the left tail head $q_i^{(tl)}$. The sign system structure sketch is defined such that the relation between this signs and q_i are preserved if a model M satisfies:

1. $M(q_i) = M(q_i^{(r)}) \otimes_I M(q_i^{(m)}) \otimes_I M(q_i^{(l)});$
2. $M(q_i^{(r)}) = M(q_i^{(hr)}) \otimes_I M(q_i^{(tr)});$
3. $M(q_i^{(l)}) = M(q_i^{(hl)}) \otimes_I M(q_i^{(tl)});$

this interpretation for signs reflect the relations between I -projections (see example 2) expressed in the following diagram:

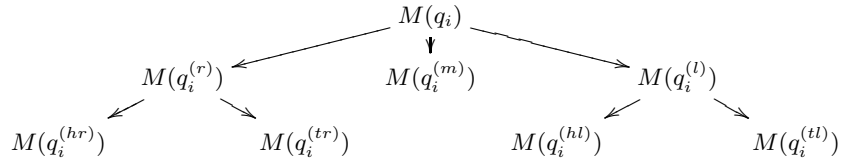


Fig. 28. Sign interpretation structure.

And models of each instruction must satisfy the following conditions:

1. For print instruction $q_0[f]q_1$ we should have;

$$M(q_i[f]t(q_j)) \circ \otimes M(q_i) \otimes M(q_i[f]t(q_j)) = M(t(q_j)) \Rightarrow \begin{cases} M(t(q_j)^{(r)}) = M(q_i^{(r)}) \\ M(t(q_j)^{(m)}) = M(f) \circ \otimes M(q_i^{(m)}) \otimes M(f) \\ M(t(q_j)^{(l)}) = M(q_i^{(l)}) \end{cases}$$

2. For instructions of type "Move one square to the left" $q_0[d_0 : L]_{\lambda} q_1$ we must have;

$$\begin{cases} M(q_i^m) \otimes M(d_0) \geq \lambda \\ M(q_i[d_0:L]_{\lambda} t(q_j))^{\circ} \otimes M(q_i) \otimes M(q_i[d_0:L]_{\lambda} t(q_j)) = M(t(q_j)) \end{cases} \Rightarrow \begin{cases} M(t(q_j)^{(r)}) = M(q_j^{(r)}) \otimes_I M(q_i^{(m)}) \\ M(t(q_j)^{(m)}) = M(q_i^{(hl)}) \\ M(t(q_j)^{(l)}) = M(q_i^{(tl)}) \end{cases}$$

3. For instructions of type "Move one square to the right" $q_0[d_0 : R]_{\lambda} q_1$ we must have;

$$\begin{cases} M(q_i^m) \otimes M(d_0) \geq \lambda \\ M(q_i[d_0:R]_{\lambda} t(q_j))^{\circ} \otimes M(q_i) \otimes M(q_i[d_0:R]_{\lambda} t(q_j)) = M(t(q_j)) \end{cases} \Rightarrow \begin{cases} M(t(q_j)^{(r)}) = M(q_i^{(hr)}) \\ M(t(q_j)^{(m)}) = M(q_i^{(tr)}) \\ M(t(q_j)^{(l)}) = M(q_i^{(ml)}) \otimes_I M(q_i^{(tl)}) \end{cases}$$

So a model M assigning to each stat a fuzzy tape with signs in F , which can be seen as a infinite chain of indexed products (see example 2):

$$t = \underbrace{\cdots \otimes_I d_5 \otimes_I d_3}_{t^{(r)}} \otimes_I \underbrace{d_1}_{t^{(m)}} \otimes_I \underbrace{d_2 \otimes_I d_4 \otimes_I \cdots}_{t^{(l)}}$$

$\begin{matrix} t^{(hr)} & t^{(tr)} & & t^{(hl)} & t^{(tl)} \\ \overbrace{\cdots \otimes_I d_5 \otimes_I d_3} & \overbrace{d_1} & \otimes_I & \overbrace{d_2 \otimes_I d_4 \otimes_I \cdots} \end{matrix}$

where we fixed a componente $t^{(m)} = d_1$ and such that each d_i is a concept description $d_i : F \times I \rightarrow \Omega$. And, the model M associates to each possible instruction (componente) a relation between fuzzy tapes t_0 and t_1 , satisfying the described proprieties.

A fuzzy Turing machine begins its execution in a initial stat and it is a parallel device, at a given moment it can assume more than a stat. It finish its execution when it is stall in a stat or set of stats.

13 Consequence relation

In a semiotic (S, M) , we define for every relation D in $Lang(S)$ the set of its λ -answers as:

$$ans_{\lambda}(D) = \{g \in M(S) : g \models_{\lambda} \forall D' D\}$$

and it can be seen as the set of concepts λ -consistent with D on the domain defined by $D' \in Lang(S)$.

Example 25 The examples presented in this section are defined using a grid semiotic, having expressive power to codify structures in a grid, using a three truth-values logic $\Omega = \{\perp, \frac{1}{2}, \top\}$.

Let D be the diagram defining a relation between pairs of entities in a grid, presented in fig. 29, where white points \bar{x} mean $M(D)(\bar{x}) = \perp$, gray points mean $M(D)(\bar{x}) = \frac{1}{2}$ and darker points \bar{x} mean $M(D)(\bar{x}) = \top$.

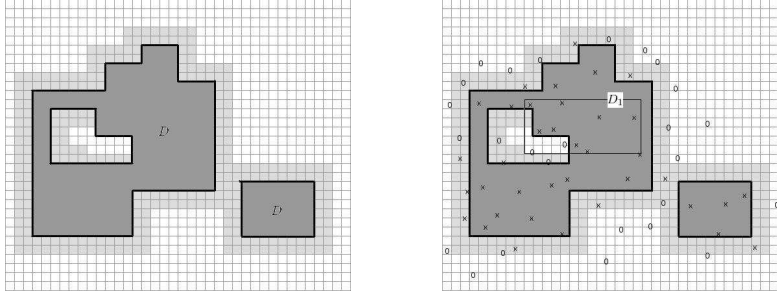


Fig. 29. Relation defined interpreting D and finite relations g_1 where a point \bar{x} marked with a X means $g_1(\bar{x}) = \top$ and a point marked with 0 means $g_1(\bar{x}) = \perp$.

The interior of the box, presented on the figure and labeled with D_1 , can be seen as the set of points described by this diagrams. The relation g_1 presented we can be seen as an example satisfying

$$g_1 \models_{\frac{1}{2}} \forall D, g_1 \models \forall D_1 D,$$

which can be expressed writing

$$g_1 \in \text{ans}_{\frac{1}{2}}(D).$$

Note what, given D the set $\text{ans}_{\lambda}(D)$ have at least an element, $M(D) \in \text{ans}_{\lambda}(D)$. Naturally:

Theorem 1 *If D is a relation in $\text{Lang}(S)$ and $\lambda_0 \leq \lambda_1$ then*

$$\text{ans}_{\lambda_1}(D) \subseteq \text{ans}_{\lambda_0}(D).$$

If $g \in \text{ans}_{\lambda}(D)$, with $g \models_{\lambda} \forall D' D$, we express this relation by writing $g_{D'} \in \text{ans}_{\lambda}(D)$.

Let D be a relation defined is a semiotic, by

$$f \leq_D g,$$

we mean that

$$\text{if } M(D)(\bar{x}) = \top \text{ then } f(\bar{x}) \leq g(\bar{x}).$$

We use this relation and the operator ans_{λ} to define two modal operators, $\diamond_{\lambda}g$ and $\Box_{\lambda}g$, as the weak and the strong images, respectively, for description $g \in M(S)$ along the relation \models_{λ} :

$$\diamond_{\lambda}g = \{D \in \text{Lang}_R(S) : (\exists f_{D'} \in \text{ans}_{\lambda}(D))(g \leq_{D'} f)\}$$

$$\Box_{\lambda}g = \{D \in \text{Lang}_R(S) : (\forall f_{D'} \in \text{ans}_{\lambda}(D))(f \leq_{D'} g)\}$$

Where $\diamond_{\lambda}g$ and $\Box_{\lambda}g$ can be seen, respectively, as the set of models λ -consistent with parts of g and the set of models λ -consistent with g in the language $\text{Lang}(S)$.

Example 26 For grid semiotic with three truth-values we presente in fig. 30 two possible diagrams $D_1 \in \diamond_{\frac{1}{2}}g_1$ and $D_2 \in \square_{\frac{1}{2}}g_2$.

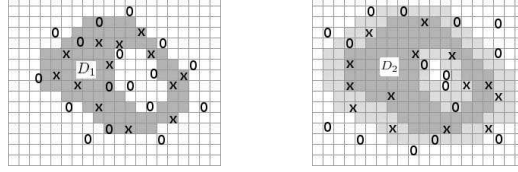


Fig. 30. $D_1 \in \diamond_{\frac{1}{2}}g_1$ and $D_2 \in \square_{\frac{1}{2}}g_2$.

We have:

Theorem 2 Given relations D_0 and D_1 in $Lang_R(S)$ and a description g :

1. if $\lambda_0 \leq \lambda_1$, then $\square_{\lambda_1}g \subseteq \square_{\lambda_0}g$,
2. if $\lambda_0 \leq \lambda_1$, then $\diamond_{\lambda_1}g \subseteq \diamond_{\lambda_0}g$,
3. if $D_0, D_1 \in \square_{\lambda}g$ then $D_0 \vee D_1 \in \square_{\lambda}g$ and
4. if $D_0, D_1 \in \diamond_{\lambda}g$ then $D_0 \wedge D_1 \in \diamond_{\lambda}g$.

In the other direction we can extend ans_{λ} to a set of relations U in $Lang(S)$:

$$ans_{\lambda}(U) = \bigvee \{g \in M(S) : (\exists D \in \square_{\lambda}g)(D \in U)\}$$

the greatest description λ -consistent with a model from U , and let

$$mod_{\lambda}(U) = \bigwedge \{g \in M(S) : (\forall D \in \square_{\lambda}g)(D \in U)\}$$

be a description λ -consistent with every model existent in U .

Theorem 3 Let U and V be sets of relations in S . Then

1. if $\lambda_0 \leq \lambda_1$, $ans_{\lambda_0}(U) \geq ans_{\lambda_1}(U)$,
2. $ans_{\lambda}(U \cup V) = ans_{\lambda}(U) \vee ans_{\lambda}(V)$, and
3. $mod_{\lambda}(U \cup V) = mod_{\lambda}(U) \wedge mod_{\lambda}(V)$.

And if $U \subseteq V$

1. $ans_{\lambda}(U) \leq ans_{\lambda}(V)$ and
2. $mod_{\lambda}(V) \leq mod_{\lambda}(U)$.

The λ -interior of concept g in the semiotic system (S, M) is defined as the greatest part of g λ -consistent with a model defined in the associated language and is given by:

$$int_\lambda(g) = \bigvee \{h \in M(S) : (\exists D \in \square_\lambda h)(\forall f_{D'} \in ans_\lambda(D))(f \leq_{D'} g)\},$$

and can be seen as the greatest fragment of g having a model λ -consistent in the language of the semiotic. It is an *interior operator* since;

1. $int_\lambda(g) \leq g$,
2. if $g \leq f$ then $int_\lambda(g) \leq int_\lambda(f)$ and
3. $int_\lambda(g) = int_\lambda(int_\lambda(g))$.

And given $\lambda_0 \leq \lambda_1$, $int_{\lambda_1}(g) \leq int_{\lambda_0}(g)$. A concept description g is called λ -open in (S, M) if

$$int_\lambda(g) = g.$$

For every set of relations U , $ans_\lambda(U)$ and $mod_\lambda(U)$ are examples of λ -opens since:

Proposition 21 *In a semiotic for every set of relations U and $\lambda \in \Omega$:*

1. $int_\lambda(ans_\lambda(U)) = ans_\lambda(U)$, and
2. $int_\lambda(mod_\lambda(U)) = mod_\lambda(U)$.

More precisely:

Theorem 4 *In a semiotic for every set of relations U ,*

$$ans_\top(U) \models \bigvee U \text{ and } mod_\top(U) \models \bigwedge U.$$

The closure of concept g in the semiotic system (S, M) is defined as the shorter cover of g codified in the language $L(S)$ and is given by:

$$cl_\lambda(g) = \bigwedge \{h \in M(S) : (\forall D \in \square_\lambda h)(\exists f_{D'} \in ans_\lambda(D))(g \leq_{D'} f)\}$$

and can be seen as the shortest cover containing g and codified in the language associated to the semiotic. It is a *closure operator* since;

1. $g \leq cl_\lambda(g)$,
2. if $g \leq f$ then $cl_\lambda(g) \leq cl_\lambda(f)$, and
3. $cl_\lambda(cl_\lambda(g)) = cl_\lambda(g)$.

And given $\lambda_0 \leq \lambda_1$, $cl_{\lambda_1}(g) \leq cl_{\lambda_0}(g)$. Trivially we have:

Proposition 22 *Given a semiotic (S, M) , for every $g \in M(S)$,*

$$int_\lambda(g) \leq g \leq cl_\lambda(g).$$

A concept description g is called λ -close in (S, M) if $cl_\lambda(g) = g$. Descriptions $ans_\lambda(U)$ and $mod_\lambda(U)$ are also λ -closed concepts. This can be extended to every λ -open description:

Proposition 23 *Given a semiotic (S, M) , for every $g \in M(S)$, g is λ -closed iff it is λ -open.*

In this sense when a description is λ -open or λ -close we called it a description λ -representable on the semiotic. By this we mean that:

Proposition 24 *Let g be a description in the semiotic (S, M) . Exists a relation D , such that $g \models_\lambda D$, iff g is λ -open or λ -close.*

Because of the symmetry between the left and the right side of $d \models D$, from the above definitions we have

$$int_\lambda = ans_\lambda \square_\lambda \text{ and } cl_\lambda = mod_\lambda \diamond_\lambda$$

and they also have symmetric definitions, obtained by replacing each operator with its symmetric:

$$\mathcal{A}_\lambda = \square_\lambda ans_\lambda \text{ and } \mathcal{C}_\lambda = \diamond_\lambda mod_\lambda.$$

By symmetry it is immediate that \mathcal{C}_λ is an interior operator and \mathcal{A}_λ is a closure operator.

Spelling out the definition of \mathcal{A}_λ , for every set of relations U ,

$$\mathcal{A}_\lambda(U) = \{D \in Lang(S) : (\forall f_{D'} \in ans_\lambda(D))(f \leq_{D'} ans_\lambda(U))\},$$

i.e. all λ -answers for D are λ -codified using relation in U . And we have:

Theorem 5 *For every pair U and V of relations in the semiotic (S, M) ,*

1. $\mathcal{A}_\lambda(U \cup V) \supseteq \mathcal{A}_\lambda(U) \cup \mathcal{A}_\lambda(V)$,
2. if $U \subseteq V$, $\mathcal{A}_\lambda(U) \subseteq \mathcal{A}_\lambda(V)$,
3. if $\lambda_0 \leq \lambda_1$, $\mathcal{A}_{\lambda_1}(U) \subseteq \mathcal{A}_{\lambda_0}(U)$,
4. if $D \in \mathcal{A}_\lambda(U)$ then $\bigvee ans_\lambda(D) \leq ans_\lambda(U)$, and
5. if $D \in \mathcal{A}_\lambda(U)$ then $ans_\lambda(U) \models_\lambda D$.

Spelling out operator \mathcal{C} we have

$$\mathcal{C}_\lambda(U) = \{D \in Lang(S) : (\exists f_{D'} \in ans_\lambda(D))(mod_\lambda(U) \leq_{D'} f)\},$$

by $D \in \mathcal{C}_\lambda(U)$ we mean that D have an λ -answer and every λ -codification for it are in U . In this case we may proof:

Theorem 6 *For every pair U and V of relations in the semiotic (S, M) ,*

$$\mathcal{C}_\lambda(U \cup V) \subseteq \mathcal{C}_\lambda(U) \cap \mathcal{C}_\lambda(V)$$

when $U \subseteq V$, $\mathcal{C}_\lambda(U) \subseteq \mathcal{C}_\lambda(V)$.

Lets write

$$U \vdash_{\lambda} D \text{ iff } D \in \mathcal{A}_{\lambda}(U),$$

and since \mathcal{A}_{λ} is a closure operator:

Theorem 7 *In a semiotic (S, M) , for every λ , we have:*

1. *if $D \in U$ then $U \vdash_{\lambda} D$ (Inclusion),*
2. *if $U \vdash_{\lambda} D$ then $U \cup V \vdash_{\lambda} D$ (Monotony), and*
3. *if $V \vdash_{\lambda} D$ and $U \cup \{D\} \vdash_{\lambda} D'$, then $U \cup V \vdash_{\lambda} D'$ (Cut).*

By this we mean that $(Lang_R(S), \vdash_{\lambda})$ is a *inference system* [26] for every λ .

Example 27 *Given interpretations, presented in fig. 31, for three diagrams D_0 , D_1 and D_2 in the grid semiotic with three valued logic:*

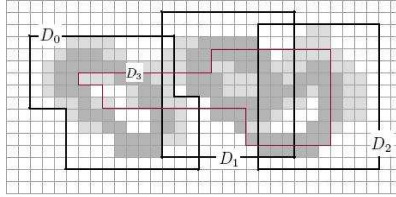


Fig. 31. Interpretations for diagrams D_0 , D_1 , D_2 and D_3 .

The diagram D_3 , with the represented interpretation, can be see as the result of applying inference to the set of diagrams $\{D_0, D_1, D_2\}$, symbolically we write

$$D_0, D_1, D_2 \vdash D_3.$$

Since $\mathcal{A}_{\lambda_0} \leq \mathcal{A}_{\lambda_1}$, if $\lambda_0 \geq \lambda_1$, we have:

$$\frac{U \vdash_{\lambda_0} D_0 \quad \lambda_0 \geq \lambda_1}{U \vdash_{\lambda_1} D_0}$$

And we have using definition of λ -consistence:

Theorem 8 *Let $g \models_{\lambda_0} \forall_{D'_0} D_0$ and $f \models_{\lambda_1} \forall_{D'_1} D_1$:*

1. $g \wedge f \models_{\lambda_0 \wedge \lambda_1} \forall_{D'_0 \otimes D'_1} D_0 \wedge D_1$,
2. $g \vee f \models_{\lambda_0 \vee \lambda_1} \forall_{D'_0 \otimes D'_1} D_0 \vee D_1$,
3. $g \otimes f \models_{\lambda_0 \otimes \lambda_1} \forall_{D'_0 \otimes D'_1} D_0 \otimes D_1$, and
4. $g \Rightarrow f \models_{\lambda_0 \Rightarrow \lambda_1} \forall_{D'_0 \otimes D'_1} D_0 \Rightarrow D_1$.

Using this properties on the definition of λ -answer we have:

Theorem 9 *On a semiotic we have:*

1. For every diagram $D \in \text{Lang}_R(S)$:
 - (a) $\text{ans}_{\lambda_0 \vee \lambda_1}(D) = \text{ans}_{\lambda_0}(D) \vee \text{ans}_{\lambda_1}(D)$,
 - (b) $\text{ans}_{\lambda_0 \wedge \lambda_1}(D) = \text{ans}_{\lambda_0}(D) \wedge \text{ans}_{\lambda_1}(D)$ and
 - (c) $\text{ans}_{\lambda_0 \otimes \lambda_1}(D) = \text{ans}_{\lambda_0}(D) \otimes \text{ans}_{\lambda_1}(D)$;
2. For every concept description $g \in M(S)$:
 - (a) $\Box_{\lambda_0 \vee \lambda_1}(g) = \Box_{\lambda_0}(g) \vee \Box_{\lambda_1}(g)$,
 - (b) $\Box_{\lambda_0 \wedge \lambda_1}(g) = \Box_{\lambda_0}(g) \wedge \Box_{\lambda_1}(g)$ and
 - (c) $\Box_{\lambda_0 \otimes \lambda_1}(g) = \Box_{\lambda_0}(g) \otimes \Box_{\lambda_1}(g)$;
3. For every set of diagrams $U \subset \text{Lang}_R(S)$:
 - (a) $\text{ans}_{\lambda_0 \vee \lambda_1}(U) = \text{ans}_{\lambda_0}(U) \vee \text{ans}_{\lambda_1}(U)$,
 - (b) $\text{ans}_{\lambda_0 \wedge \lambda_1}(U) = \text{ans}_{\lambda_0}(U) \wedge \text{ans}_{\lambda_1}(U)$ and
 - (c) $\text{ans}_{\lambda_0 \otimes \lambda_1}(U) = \text{ans}_{\lambda_0}(U) \otimes \text{ans}_{\lambda_1}(U)$.

Which gives support to the definition of the introduction rules:

$$\frac{U \vdash_{\lambda_0} D_0 \quad U \vdash_{\lambda_1} D_1}{U \vdash_{\lambda_0 \vee \lambda_1} D_0 \vee D_1, U \vdash_{\lambda_0 \wedge \lambda_1} D_0 \wedge D_1, U \vdash_{\lambda_0 \otimes \lambda_1} D_0 \otimes D_1}$$

The fact of, if $D_0 \wedge D_1 \in \mathcal{A}_\lambda(U)$ then $D_0 \in \mathcal{A}_\lambda(U)$ and $D_1 \in \mathcal{A}_\lambda(U)$, can be expressed by the elimination rule:

$$\frac{U \vdash_\lambda D_0 \wedge D_1}{U \vdash_\lambda D_0, U \vdash_\lambda D_1}$$

Naturally, in a divisible logic, we have

$$\frac{U \vdash_{\lambda_0} D_0 \quad U \vdash_{\lambda_1} D_0 \Rightarrow D_1}{U \vdash_{\lambda_0 \otimes \lambda_1} D_0 \wedge D_1}$$

since if $g \models_{\lambda_0} \exists D_0$ and $g \models_{\lambda_1} \exists(D_0 \Rightarrow D_1)$ then $g \models_{\lambda_0 \otimes \lambda_1} \exists D_1$. Because, if $D_0 \in \mathcal{A}_{\lambda_0}(U)$ and $D_0 \Rightarrow D_1 \in \mathcal{A}_{\lambda_1}(U)$ then $D_0 \wedge D_1 \in \mathcal{A}_{\lambda_0 \otimes \lambda_1}(U)$. By this we mean what for every $f \in \text{ans}_{\lambda_0}(D_0)$, $f \leq \text{ans}_{\lambda_0}(U)$, and for every $h \in \text{ans}_{\lambda_0}(D_1)$, $f \Rightarrow h \in \text{ans}_{\lambda_1}(D_0 \Rightarrow D_1)$, and $f \otimes (f \Rightarrow h) \in \text{ans}_{\lambda_0 \otimes \lambda_1}(U)$. Note that, in a divisible ML-algebra, $f \otimes (f \Rightarrow h) \leq f \wedge h$. Then $f \wedge h \in \text{ans}_{\lambda_0 \otimes \lambda_1}(U)$.

A diagram D codifies all the information existent in a concept d , using the syntax associated to semiotic (S, M) , if for every diagram D_1 such that $d \models_\lambda \forall D_1$, $M(D \vee D_1) = M(D)$. This diagrams are called *total* and can be defined by

$$D = \bigvee_{d \models_\lambda \forall D_i} D_i.$$

In the category $\text{Lang}(S)$ having by objects diagrams codifying relations and where a diagram D is a morphism from relation D_0 to relation D_1 if $D_0 \otimes D = D_1$, we consider composition as the operations of diagram gluing. The consequence operator \vdash_λ can be seen as a functor:

$$\vdash_\lambda: \text{Lang}(S) \rightarrow \text{Lang}(S).$$

A diagram D is called a *theory* in the λ -semiotic (S, M) if it is a fixed-point for consequence operator

$$\vdash_{\lambda}(D) = D.$$

The semiotic model M , can be interpreted as a functor

$$M : \text{Lang}(S) \rightarrow \lambda\text{-Hy}_{(S, M)},$$

in the category of concepts λ -consistentes and computable multi-morphisms. A functor in the opposite direction can be defined using the operator of consistence

$$\models_{\lambda} : \lambda\text{-Hy}_{(S, M)} \rightarrow \text{Lang}(S),$$

assigning to each λ -consistente description a total diagram with its codification on the semiotic.

Since $M(D) \models_{\lambda} \forall D$ we have

$$M \circ \models_{\lambda} = id,$$

and by definition of consequence relation

$$\models_{\lambda} \circ M = \vdash_{\lambda}.$$

If D is a λ -theory in the semiotic, $M(D)$ is the model λ -consistent with this theory.

14 Integration

Our aim is to construct an integration semiotic base from several separated semiotics. This need can arise, for example, when knowledge bases are acquired independently from interactions with several domain experts. A similar problem can also arise whenever separated knowledge bases are generated by learning algorithms. The objective of integration is then construct one system that exploits all the knowledge that is available and has a good performance, i.e. a good degree of consistence with the data.

We must differentiate between two types of integration: semiotics integration and integration of models in a semiotic. The semiotic integration goal is the definition of a semiotic integrating the sintaxe and semantic of a given family of semiotics. By the integration of models in a semiotic we mean the possibility of improve the description of concepts integrating models for it using diferente data or diferente views of the same data. The integration of models is defined by an integration schema describing the relations between different models in the same semiotic. In the semiotic integrating we integrate different logics in the same semiotic associated to different languages used by domain experts or associated to structure specification language. In both senses Knowledge integration, in conjunction with inference, can play an important rule in the process of knowledge acquisition.

We impose an important restriction to the semiotic integration: Given a family of semiotics $(S_i, M_i)_I$ its integration is defined, if and only if, equal signs and components with the same label in different semiotics have the same interpretation, with only a possible exception, the interpretations of sign Ω associated to the semiotics logic and its operators may be different.

The integration of semiotics $(S_i, M_i)_I$ is denoted by $(\bigcup_I S_i, \bigcup_I M_i)$ and it is given by the sign system

$$\bigcup_I S_i = (\bigcup_I L_i, \bigcup_I \mathcal{E}_i, \bigcup_I \mathcal{U}_i, \bigcup_I \text{col}\mathcal{U}_i),$$

if, for each $i \in I$, the semiotic S_i is defined by the structure $(L_i, \mathcal{E}_i, \mathcal{U}_i, \text{col}\mathcal{U}_i)$. Where $\bigcup_I L_i$ is the library defined by the union of libraries $(L_i : |L_i| \rightarrow (\text{Chains} \downarrow \Sigma_i^+))_I$ associated to each semiotic. This library is given by

$$\bigcup_I L_i : \bigcup_I |L_i| \rightarrow (\text{Chains} \downarrow \bigcup_I \Sigma_i^+),$$

having by signs the union of ontology $\bigcup_I \Sigma_i^+$ defined by the signs existent in each library, and having by component labels the union of labels existent in both libraries. Note that the integration of libraries must preserve component functionalities. In this sense, the union of libraries is only defined if the component existent in different libraries, with equal label, have the same functionalities. The graphic language associate to $\bigcup_I L_i$ is denoted by $\text{Lang}(\bigcup_I L_i)$, and we have $\bigcup_I \mathcal{E}_i \subset \text{Lang}(\bigcup_I L_i)$.

From the description for the language associated to $\bigcup_I L_i$ recall what: Given two graphs G_0 and G_1 we define $G_0 \bigcup G_1$ as the graph defined having by vertices the vertices of G_0 and G_1 and having by arrows the arrows of G_0 and G_1 . If each library L_i have associated multi-graphs $\mathcal{G}(L_i)$, we have

$$\mathcal{G}(\bigcup_I L_i) = \bigcup_I \mathcal{G}(L_i).$$

Then, if we have models $(M_i : \mathcal{G}(L_i^*)) \rightarrow \text{Set}(\Omega)_I$ for different libraries, the homomorphism $\bigcup_I M_i$ is a model for the sign system $\bigcup_I S_i$,

$$\bigcup_I M_i : \mathcal{G}((\bigcup_I L_i)^*) \rightarrow \text{Set}(\Omega)$$

constructed using the union of models $(M_i : \mathcal{G}(L_i^*)) \rightarrow \text{Set}(\Omega)_I$, making for nodes $\bigcup_I M_i(v) = M_i(v)$ if $v \in \mathcal{G}(L_i^*)$ and $v \neq \Omega$, for some $i \in I$, and $\bigcup_I M_i(v) = M_i(v)$ for multi-arrows $f : w \rightarrow w' \in \mathcal{G}(L_i^*)$, and $w' \neq \Omega$ for some $i \in I$. By this we mean that not logic signs and multi-arrows which not represent relations are interpreted as it where in its libraries. This definition only makes sense when equal signs and equal labels have equal interpretations in different libraries.

For the logic family of logic signs $(\Omega_i)_I$ associated to the family of logic semiotics $(S_i)_I$ we define the sign $\prod_I \Omega_i$ interpreted by $\bigcup_I M_i$ as the ML-algebra product $\prod_I M_i(\Omega_i)$. The interpretation of sign $\prod_I \Omega_i$ is a ML-algebra and for

every relation $r : w \rightarrow \Omega_i$ existent in each semiotic S_i its interpretation by $\bigcup_I M_i$ is the relation

$$\bigcup_I M_i(r : w \rightarrow \Omega_i) = M_i(r) \otimes \pi_{\Omega_i}^\top$$

where $\pi_{\Omega_i}^\top : \Omega_i \rightarrow \prod_I \Omega_i$ is the map such that

$$\pi_{\Omega_i}^\top(\alpha) = (\top, \dots, \top, \alpha, \top, \dots, \top),$$

having different of \top only the component of order i . Formally,

Proposition 25 *If*

$$S_0 = (L_0, \mathcal{E}_0, \mathcal{U}_0, \text{col}\mathcal{U}_0), S_1 = (L_1, \mathcal{E}_1, \mathcal{U}_1, \text{col}\mathcal{U}_1), \dots, S_n = (L_n, \mathcal{E}_n, \mathcal{U}_n, \text{col}\mathcal{U}_n)$$

are sign systems with models M_0, M_1, \dots, M_n then

$$\bigcup_I M_i : \mathcal{G}((\bigcup_I L_i)^*) \rightarrow \text{Set}(\Omega)$$

defined as above is a model for the sign system

$$\bigcup_I S_i = (\bigcup_I L_i, \bigcup_I \mathcal{E}_i, \bigcup_I \mathcal{U}_i, \bigcup_I \text{col}\mathcal{U}_i).$$

Since, for models M_0, M_1, \dots, M_n of sign system S_0, S_1, \dots, S_n , we have, by definition 15, for every $j = 1, \dots, n$:

1. if $D \in \mathcal{E}_j$, then $\bigcup_I M_i(D) = M_j(D)$ is a total multi-morphism;
2. if $(s, D, i(D), o(D)) \in \mathcal{U}_j$, then $\bigcup_I M_i(s) = M_j(s)$ is the Ω -set defined by $\text{Lim } MD$;
3. if $(s, D, i(D), o(D)) \in \text{col}\mathcal{U}_j$, then $\bigcup_I M_i(s) = M_j(s)$ is the Ω -set defined by $\text{coLim } MD$.

Naturally, the resulting semiotic of an integration process have the syntax and the semantic generated by the syntax and semantic associated to the semiotics. The some principle can be seen for some syntactic operators. The integration of a family of semiotics, where at last one is a differential semiotics, is a diferencial semiotic and the same happens for temporal semiotics. If $(S_i)_I$ is a family where $(S_j)_J$ is a subfamily of temporal semiotics, given by syntactic operator $t_i : \Sigma_i^+ \rightarrow \Sigma_i^+$. Then the integration $\bigcup_I S_i$ is a temporal semiotic where the syntactic operator $t : \bigcup_I \Sigma_i^+ \rightarrow \bigcup_I \Sigma_i^+$ is defined by making:

1. $t(s) = t_j(s)$ if $s \in \Sigma_j^+$ where $j \in J$, and
2. $t(s) = s$ if $s \notin \bigcup_J \Sigma_j^+$.

An integration schema is a diagram

$$\mathcal{J} : \mathcal{G} \rightarrow \text{Set}(\Omega),$$

defined on the category of interpretations and computable multi-morphisms, such that

$$\mathcal{J}(i) = MD_i$$

for every vertices i in \mathcal{J} . Let (D_i) be a family of diagrams used on integration schema \mathcal{J} definition. The concept description $\Omega(\mathcal{J})$ defined by \mathcal{J} is given by the colimit of \mathcal{J}

$$\Omega(\mathcal{J}) = \text{colim}_i MD_i = \text{colim}_i \text{Lim } M(D_i),$$

where *colim* is computed as defined in 11, i.e.

$$(\text{coLim } \mathcal{J})(\dots, \bar{x}_i, \dots, \bar{x}_j, \dots) = \dots \otimes \text{Lim } M(D_i)(\bar{x}_i) \otimes \dots \otimes \text{Lim } M(D_j)(\bar{x}_j) \otimes \dots \otimes \bigvee_{f: MD_i \rightarrow MD_j \in \mathcal{J}} f(\bar{x}_i, \bar{x}_j).$$

15 Reasoning about models of concepts

The language $\lambda\text{-RL}(S)$ of λ -representable logic is a formalism to speak of structures λ -representable on a semiotic (S, M) . It is basically a classic string-based modal logic defined by a generative grammar where propositional variables are interpreted as diagrams belonging to the language associated to the sign system S .

$\lambda\text{-RL}(S)$ is constructed from relations in $\text{Lang}(S)$, modal operators limit, closure, interior and the lifting of the monoidal logic connectives \otimes , \Rightarrow , \wedge and \vee to relations.

Every semiotic (S, M) defines a semantic for $\lambda\text{-RL}(S)$ by the truth-relation

$$g \models_{\lambda} \varphi,$$

given, for every formula $\varphi \in \lambda\text{-RL}(S)$ and every concept description g in (S, M) , as follows:

1. $g \models_{\lambda} \varphi$ iff φ is the diagram D and $F(g, MD) \geq \lambda$,
2. $g \models_{\lambda} [I]\varphi$ iff $\text{int}(g) \models_{\lambda} \varphi$,
3. $g \models_{\lambda} [C]\varphi$ iff $\text{cl}(g) \models_{\lambda} \varphi$.

And given formulas φ_0 and φ_1 in $\lambda\text{-RL}(S)$, if

$$g \models_{\lambda_0} \varphi_0 \text{ and } g \models_{\lambda_1} \varphi_1$$

we have:

1. $g \models_{\lambda_0 \otimes \lambda_1} (\varphi_0 \otimes \varphi_1)$,
2. $g \models_{\lambda_0 \Rightarrow \lambda_1} (\varphi_0 \Rightarrow \varphi_1)$,
3. $g \models_{\lambda_0 \wedge \lambda_1} (\varphi_0 \wedge \varphi_1)$ and
4. $g \models_{\lambda_0 \vee \lambda_1} (\varphi_0 \vee \varphi_1)$.

Using the structural compatibility between multi-morphism composition and diagram gluing we have:

Proposition 26 *Given multi-morphism g and h such that*

$$g \models_{\lambda_0} \varphi_0 \text{ and } h \models_{\lambda_1} \varphi_1$$

we have:

$$g \otimes h \models_{\lambda_0 \otimes \lambda_1} \varphi_0 \otimes \varphi_1$$

By the lifting of the ML-algebra structure to the set of concept descriptions we have:

Proposition 27 *Given concept descriptions g and h in $\oplus_i A_i$ such that $g \models_{\lambda_0} \varphi$ and $h \models_{\lambda_1} \varphi$ we have:*

1. $(g \otimes h) \models_{\lambda_0 \otimes \lambda_1} \varphi$,
2. $(g \Rightarrow h) \models_{\lambda_0 \Rightarrow \lambda_1} \varphi$,
3. $(g \wedge h) \models_{\lambda_0 \wedge \lambda_1} \varphi$ and
4. $(g \vee h) \models_{\lambda_0 \vee \lambda_1} \varphi$.

Given a set of relations U from $Lan_R(S)$ and φ a relation in $\lambda\text{-RL}(S)$ we define:

$$U \models_{\lambda} \varphi \text{ iff } ans_{\lambda}(U) \models_{\lambda} \varphi$$

Using theorem 5 we have:

Theorem 10 (Soundness) *Given a set of relations U from $Lang_R(S)$ and φ a relation in $\lambda\text{-RL}(S)$,*

$$\text{if } U \vdash_{\lambda} \varphi \text{ then } U \models_{\lambda} \varphi$$

for every λ .

Naturally, the completeness isn't valid, if $U \models_{\lambda} \varphi$, we may not prove using deduction $U \vdash_{\lambda} \varphi$.

16 Conclusions and future work

The use of semiotics seems to be the appropriate formalism for defining syntax and the meaning of graphic language. Particularly when this languages are based on a library of functional components interpreted as relations evaluated in a multi-valued logic. This approach makes simplifies the integration of knowledge expressed using different languages and allowing the ingerence of new knowledge.

References

- [1] A. Birman, *The tmg recognition schema*, Doctoral thesis, Princeton University, Dept. of Electronic Engineering, 1970.
- [2] C. Bishop, *Neural network for pattern recognition*, Oxford University Press, 1996.
- [3] B. Borceux, *Handbook of categorical algebra 1: Basic category theory*, Cambridge University Press, 1994.

- [4] J. Bosch, *Superimposition: a component adaptation technique*, Information and Software Technology , v.41 n.5, p.257-273, Elsevier, 1999.
- [5] N. Chomsky, *Syntactic structures*, The Hague: Mouton, 1957.
- [6] A. Wolf D. Perry, *Foundations for the study of software architectures*, ACM SIG-SOFT Software Engineering Notes , v.17 n.4, p.40-52, ACM, 1992.
- [7] D. Temperley D. Sleator, *Parsing english with a link grammar*, Technical report CMU-CS-91-196, Carnegie Mello University, School of Computer Science, 1991.
- [8] J. Castro F. Klawonn, *Similarity in fuzzy reasoning*, Mathware and soft comput., v.2, p.197-228, 1995.
- [9] B. Ford, *Parsing expression grammars: A recognition-based syntactic fundation.*, In *ACM SIGPLAN Notices*, v.39 n.1, p.111-122, ACM New York,USA, 2004.
- [10] J. Goguen, *An introduction to algebraic semiotics, with application to user interface design*, Lecture Notes in Artificial Intelligence, v.1562, p.242-291, 1999.
- [11] J. Rosický J. Adámek, *Locally presentable and accessible cateories*, Cambridge University Press, Cambridge, 1994.
- [12] A. Lopes J. Fiadeiro, *Semantics of architectural connectors*, TAPSOFT'97 LNCS, v.1214, p.505-519, Springer-Verlag, 1997.
- [13] R. Burstall J. Goguen, *Introducing institutions.*, Proceedins of the Carnegie Mellon Workshop on Logic of Programing, p.221-256, June 06-08, 1983.
- [14] P. Johnstone, *Sketches of an elephant: A topos theory compendium*, Oxford University Press, Oxford, 2002.
- [15] R. Kasman L. Bass, P. Clements, *Software architecture in practice*, Addison Westey, 1998.
- [16] R. Par M. Makkai, *Accessible categories: The foundations of categorical model theory*, Contemporary Matematics 104, American Mathematical Society, 1989.
- [17] S. MacLane, *Categories for working mathematician*, Springer-Verlag, 1971.
- [18] J.B. Whitham M.J. Lighthill, *On kinematic wave*, Proc. Royal Soc. London, Ser. A , v.299, p.281-345, 1955.
- [19] W. Tholen M.M. Clementino, D. Hofmann, *One setting for all: Metric, topology, uniformity, approach structure.*, Theory Appl. Categ., v.11 n.15, p.337-352, 2003.
- [20] H. Ehrig R. Heckel P. Baldan, A. Corradini, *Compositional semantics for open petri nets base deterministic processes*, Math. Struct. in Comp. Science, v.15, p.1-35, 2006.
- [21] D. Hand P. Cohen, N. Adams, *The ida'01 robot data challenge*, In Lecture Notes in Computer Acience, v.2189, p.87-109, Springer, Berlin/Heidelberg, 2001.
- [22] J.R. Quinlan, *Combining instance-based and model-based learning*, In procceding ML'93 (Otgoff, Ed.) Morgan Kaufmann, 1993.
- [23] D. Garlan R. Allen, *A formal basis for architectural connection*, ACM TOSEM , v.6 n.3, p.213-249, New York, 1997.
- [24] J. Gibbons R. Backhouse, *Generic programming: Advanced lectures*, Springer, 2003.
- [25] P.I. Richards, *Shock wave on the highway*, Oper. Res., v.4, p.42-51, 1956.
- [26] T. Maibaum S. Abramsky, D. Gabbay, *Handbook of logic in computer science, vol. 1*, Clarendon Press, Oxford, 1992.
- [27] R. Michalski T. Michell, J. Carbonell, *Machine learning: A guide to current research*, The Springer International Series in Engineering and Computer Science, Springer, 1986.
- [28] L. Tesnière, *Eléments de syntaxe structures*, Kliencksieck, Paris, 1959.
- [29] L. Valverde, *On the structure of f-indistinguishability operators*, Fuzzy Sets and Systems, v.17, p.313-328, 1985.

- [30] WMP van der Aalst, *The application of petri nets to workflow management*, The Journal of Circuits, Systems and Computers, v.8 n.1, p.21-66, 1998.
- [31] ———, *Interorganizational workflows: An approach based on message sequence charts and petri nets*, System Analysis and Modeling, v.34 n.3, p.335-367, 1999.
- [32] F. Pissen Z. Diskin, B. Kadish, *Humans, computers, specifications: The arrow logic of information systems engineering*, Int. J. of Computing Anticipatory Systems, v.3, p.31-51, CHAOS, 1999.

